251233

XML

BIBL. CENTR. UNIV.
„M. EMINESCU" IAŞI
INFORMATICĂ
III - 1220

Lee Anne Phillips

XML

Traducere de Ioan Bledea
și Eugen Șipoș

130117
B.C.U. - IASI

Teora

Titlul original: **Special Edition Using XML**

• **Copyright © 2001 Teora**

Toate drepturile asupra versiunii în limba română aparţin Editurii **Teora**. Reproducerea integrală sau parţială a textului sau a ilustraţiilor din această carte este posibilă numai cu acordul prealabil scris al Editurii **Teora**.

Authorized translation from the Enghlish language edition entitled Special Edition Using XML, published by QUE Corporation.
Copyright © 2000
All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.
Romanian language edition published by Editura Teora Publishing [translator].
Copyright © 2000

Teora
Calea Moşilor nr. 211, sector 2, Bucureşti
Fax: 01/210.38.28
e-mail: teora@teora.kappa.ro

Teora – Cartea prin poştă
CP 79-30, cod 72450 Bucureşti, Romania
Tel: 01/252.14.31
e-mail: cpp@teora.kappa.ro

Copertă: Gheorghe Popescu
Tehnoredactare: Marius Stuparu
Director Editorial: Diana Rotaru

Preşedinte: Teodor RĂDUCANU

NOT 4845 CAL XML, TOTUL DESPRE
ISBN: 973-20-0604-8

Printed in Romania

Cuprins

PARTEA I Fundamentele XML

1 Introducere în XML 12
2 Sintaxa XML 30
3 Definiţiile tipurilor documentului XML 61
4 Extinderea definiţiei tipului documentului prin modificări locale 93
5 Crearea unui DTD pornind de la zero 111
6 Zone de nume XML 129
7 Scheme XML 143
8 XPath 169
9 XLink şi XPointer 183

PARTEA A II-A Manipularea XML

10 DOM (Document Object Model) – Modelul document obiect 206
11 SAX (the Simple API for XML) – API simplu pentru XML 226
12 CSS1, CSS2, DSSSL, XSL 257
13 Foi de stil în cascadă şi XML/XHTML 283
14 Utilizarea generaţiei următoare de foi de stil extensibile – XSL 304

PARTEA A III-A Integrarea XML cu alte tehnologii

15 Obţinerea informaţiilor despre utilizator 332
16 Securitate şi confidenţialitate 347
17 Utilizarea facilităţilor server-side – Java 379
18 Utilizarea facilităţilor server-side – ActiveX 405
19 Utilizarea brokerilor de obiecte comune 423
20 Planificarea resurselor pentru întreprinderi în XML 448

PARTEA A IV-A Alte aplicaţii ale XML

21 RDF (The Resource Description Framework – cadrul descrierii resurselor) 460
22 XHTML în detaliu 481
23 Utilizarea SMIL (The Synchronized Multimedia Integration Language – limbajul de integrare multimedia sincronizat) 500
24 Utilizarea MathML (The Math Markup Language – limbajul de marcare matematic) 539
25 Accesibilitate 582
26 Cooperarea în domeniu şi resurse 610
27 Concluzii – viitorul în Web 626

Anexa A Referinţe XML/XHTML 639
Anexa B Instrumente pentru editarea şi transformarea XML/XHTML 678
Anexa C Entităţi caracter 687
Anexa D Referinţe CSS1, CSS2 709
Glosar 750
Index 762

Despre autoare

Lee Anne Phillips a fost interesată de prelucrarea datelor și de rețele încă din fericita zi în care, la primul curs de programare, la UC Berkeley, profesorul a convins-o că are talent pentru așa ceva și ar trebui să renunțe la planurile sale pentru o carieră în psihologie lingvistică și, în schimb, să „aranjeze" biți. De la UCB a trecut prin mai multe stadii ca programator de sisteme mainframe, proiectant firmware, administrator de rețea, arhitect software, manager de proiect și programare și, în final, consultant, până când a simțit nevoia să împărtășească toate acestea cuiva. Locuiește în zona Golfului San Francisco – locul ei de naștere – este licențiată în informatică și are un deosebit simț al umorului. Puteți vizita pagina sa Web, la `http://www.leeanne.com/`.

INTRODUCERE

În această introducere

Cui îi este adresată cartea
Structura cărții
Convenții utilizate în carte

Această carte este dedicată înțelegerii noianului de standarde și propuneri concurente (și uneori contradictorii) legate de XML care plutesc pe marea posibilităților oferite de XML. Multe dintre facilitățile cele mai dorite de utilizator (dumneavoastră, stimate cititor) sunt furnizate cu ajutorul a mai mult de șase „standarde" diferite, cu grade de suport variate, provenind numai de la o mână de furnizori. Este suficient pentru a vă face să vă smulgeți părul din cap.

Dar conceptele XML sunt aproape la fel de simple ca și când ați lua un creion roșu și ați face adnotări la un text în timp ce îl citiți. Spre deosebire de HTML, care are un vocabular foarte restrâns, XML vă oferă o libertate aproape nelimitată pentru a vă descrie documentele în modul dorit de dumneavoastră.

XML devine rapid limbajul comun al Web-ului, noi furnizori alăturându-se zilnic grupării pline de succes a XML. Această carte este proiectată pentru a vă ajuta să vă alăturați lor.

Multe corporații importante, cu resurse semnificative, au investit deja în XML, printre care sunt și unii dintre greii din domeniu. Microsoft, IBM, SAP, Netscape, Oracle, Sun Microsystems, guvernul SUA, armata SUA – lista seamănă cu topul celor mai bogate 5000 de companii și agenții guvernamentale majore pe care puțini dintre noi își pot permite să le ignore, chiar dacă nu avem de a face cu ele în mod direct.

Suntem într-un moment critic în dezvoltarea XML; standardele sunt propuse și promulgate atât de repede încât nimeni nu le mai poate urmări. Există uneori propuneri de la membrii Consorțiului World Wide Web (W3C), alternative venite de la persoane fără nici o afiliere, inițiative ale unor grupuri de utilizatori și încercări de a le împăca pe toate. Această carte vă ajută să ordonați toate acestea, vă spune unde să căutați mai multe informații care să vă ajute să țineți pasul cu noile dezvoltări pe măsura apariției acestora și vă ajută la construirea unei scheme conceptuale pentru a vă permite să încadrați noile informații într-o structură existentă, pe măsură ce veți înainta. Distracție plăcută.

În prezent, toți cei care sunt on-line depind de W3C. Crearea de aplicații Web care să funcționeze este suficient de dificilă chiar dacă nu se iau în calcul sutele de sisteme particulare provenite de la diverși producători individuali. Două abordări principale sunt deja prea multe. Asemenea instrumentelor metrice și SAE, mesele noastre de lucru sunt suficient de aglomerate cu componentele necesare chiar fără să ne mai batem capul dacă pentru o piuliță avem nevoie de o cheie fixă de 19 mm sau de una de 3/4 inci.

Prin promulgarea de standarde, W3C asigură existența unor niveluri de interoperabilitate de bază între aplicațiile Web. Producătorii care se abat de la standarde riscă să pară ridicoli până la urmă, oricât de mult ar încerca să își impună deciziile sau propriile standarde asupra celorlalți.

Deci XML și standardele legate de acesta sunt toate compromisuri între unele metodologii ideale, platonice, și încercarea curajoasă de a supraviețui pe Web. Multe grupuri de lucru din cadrul W3C au avut uneori trasee diferite introducând incompatibilități sau inconsecvențe chiar în standarde, iar acestea au fost sau nu eliminate la analiza finală. Între timp, noi trebuie să perseverăm și să facem aplicațiile Web să funcționeze. Ele trebuie să funcționeze și asta vor face. Deci haideți să ne apucăm de lucru. După ce vom lua viteză, vom fi gata pentru o călătorie palpitantă.

Cui îi este adresată cartea

Această carte este destinată proiectanților Web profesioniști, programatorilor, specialiștilor în baze de date și conținut și tuturor celor implicați în publicarea sau distribuirea de informații prin orice tip de rețea. XML și standardele legate de acesta oferă comunității Web un mod valoros și flexibil de organizare și partajare a datelor. Iată câteva comunități de utilizatori potențiali

- Furnizori și autori de conținut
- Utilizatori de baze de date
- Programatori
- Proiectanți Web
- Oameni de știință și studenți
- Cercetători și analiști
- Furnizori de motoare de căutare și indexare
- Oricine dorește să învețe mai multe despre XML și viitorul Web-ului

Structura cărții

Cartea este formată din șase secțiuni importante:

■ Partea I, „Fundamentele XML", vă prezintă conceptele și facilitățile necesare utilizării eficiente a XML și a standardelor legate de acesta.

■ Partea a II-a, „Manipularea XML", examinează instrumentele utilizate pentru controlul structurii și aspectului documentului XML în sine, modelul document obiect (Document Object Model), foile de stil (Style Sheets) și SAX (the Simple API for XML - API simplu pentru XML).

■ Partea a III-a, „Integrarea XML cu alte tehnologii", studiază amănunțit aplicațiile cu baze de date și server-side pentru care XML se potrivește perfect. Exemple din lumea

reală prezintă aspectele esențiale ale aplicațiilor care funcționează în mediul de afaceri de azi. În plus, problemele importante ale confidențialității și securității on-line sunt discutate în contextul practicilor și problemelor actuale din domeniu.

■ Partea a IV-a, „Alte aplicații ale XML", completează golurile cu discuții despre modul de integrare în XML a documentelor Web existente, prin utilizarea XHTML, despre modalitățile de a crea vocabulare descriptive standard utilizând cadrul de descriere a resurselor (Resource Description Framework) și limbaje multimedia și științifice care arată modul în care XML și standardele legate de acesta pot simplifica sarcinile existente sau să facă posibile interacțiuni noi la care nici nu ați visat. În încheiere veți cerceta cum poate XML să facă Web-ul mai accesibil pentru oricine și veți vedea încotro este posibil să se îndrepte dezvoltarea Web în viitorul apropiat.

■ Anexele oferă o referință rapidă, aflată la îndemână, despre fundamente demontate până la cele mai elementare părți, alcătuind astfel un memorator rapid referitor la modul de realizare a unei anumite acțiuni sau la aspectul unei anumite caracteristici.

Convenții utilizate în carte

Această carte utilizează convenții speciale pentru a vă ajuta să beneficiați la maximum de ea și de XML.

Convenții de text

Tipurile diferite de caractere din această carte identifică termeni și alte obiecte speciale. Aceste tipuri de caractere speciale includ următoarele:

Tip	Semnificație
Cursiv	Termeni noi sau expresii definite pentru prima dată. Un termen scris cursiv și urmat de un număr de pagină indică pagina la care termenul respectiv a fost definit pentru prima dată.
`Monospatiat`	Codul, ieșirile și adresele Web apar cu acest font, indicând faptul că acestea sunt lucruri care fie vor fi introduse în calculator, fie vor fi afișate pe ecran.
`Monospatiat aldin`	Anumite intrări pe care se presupune că utilizatorul trebuie să le introducă apar cu acest font.

Elemente speciale

Sugestie de la LeeAnne.com Words to weave by

Sugestiile sunt concepute pentru a sublinia caracteristici, neplăceri şi trucuri din domeniu, pe care altfel s-ar putea să le omiteţi. Acestea vă vor ajuta să scrieţi în XML repede şi eficient.

Notă

Notele subliniază elemente cărora trebuie să le acordaţi atenţie, altfel este posibil să treceţi peste ele atunci când vă grăbiţi. În general, notele reprezintă pentru dumneavoastră un mod de a obţine nişte informaţii suplimentare despre un subiect, fără să vă obosiţi.

Atenţie

Fiţi atenţi la paragrafele „Atenţie"! Vă pot salva de la ore preţioase de muncă în zadar. Să nu spuneţi că nu v-am avertizat.

De asemenea, fiecare capitol se încheie cu o secţiune „Depanare" sau „Studii de caz". O secţiune „Depanare" vă ajută să depăşiţi problemele uzuale în utilizarea XML. O secţiune „Studii de caz" vă arată modul în care subiectul capitolului se aplică dezvoltării XML zilnice sau furnizează informaţii suplimentare interesante sau utile.

Partea I

Fundamentele XML

1 Introducere în XML
2 Sintaxa XML
3 Definiţiile tipurilor documentului XML
4 Extinderea definiţiei tipului documentului prin modificări locale
5 Crearea unui DTD pornind de la zero
6 Zone de nume XML
7 Scheme XML
8 XPath
9 XLink şi XPointer

Introducere în XML

CAPITOLUL 1

În acest capitol

Cu XML totul devine posibil
Limitările HTML
Redefinirea modalităților de dezvoltare Web
Relația între SGML, XML și XHTML
XML în teorie și practică
Exemple practice în XML
Dezvoltări viitoare
O descriere concisă a originilor XML

Cu XML totul devine posibil

Comunitatea Internet investește o cantitate enormă de energie, bani și efort în dezvoltarea unui set vast de standarde legate în jurul XML (Extensible Markup Language – limbaj de marcare extensibil), generația următoare de metode de transmitere a documentelor pe Web. Numai în 1999 au fost publicate sau propuse mai multe standarde și proiecte de standarde, aproape toate legate de XML, decât în toată istoria Consorțiului World Wide Web (W3C), grupul responsabil cu standardele Web. În 2000, vor fi publicate mai multe zeci de standarde legate de XML, dublând numărul de standarde W3C și extinzând puterea de coeziune a XML în toate colțurile World Wide Web.

XML și standardele legate de acesta vă permit să înlocuiți sau să extindeți sistemele de etichetare brevetate, cum sunt Cold Fusion de la Allaire și Active Server Page (ASP) de la Microsoft, cu limbaje independente de platformă care se potrivesc perfect cu problema spațiului din pagina dumneavoastră. În loc de (sau pe lângă) inserarea de etichete speciale sau comentarii care să explice semnificația unui anumit câmp, câmpul în sine poate deveni mai sugestiv atât pentru aplicații, cât și pentru cititori. Deci, o listă de prețuri cu comentarii, care în HTML poate arăta astfel:

```
<!-- Lista de preturi pentru anumite fructe -->
<dl>
  <!-- Fruct -->
  <dt>Mere</dt>
    <!-- Pret -->
    <dd>1$</dd>
  <!-- Fruct -->
  <dt>Portocale</dt>
    <!-- Pret -->
    <dd>2$</dd>
</dl>
```


poate fi făcută să arate astfel:

```
<ListaPreturiFructe>
  <Fruct>Mere</Fruct>
    <Pret>1$</Pret>
  <Fruct>Portocale</Fruct>
    <Pret>2$</Pret>
</ListaPreturiFructe>
```

Mai sus avem un mic exemplu al accesului mai ușor la date care poate fi realizat prin utilizarea XML. Nu numai că informația este mai puțin dezordonată și mai clar prezentată, dar și câmpurile pot fi identificate de către un motor de căutare. Deci, merele de mâncat pot fi deosebite rapid de Marele Măr (orașul New York) sau de mărul discordiei. În timp ce în HTML trebuia să potrivim datele în patul procustian al unei definiții de listă HTML pentru ca aceasta să aibă aspectul dorit de noi, în XML putem lăsa structura de date să izvorască chiar din date, iar pentru formatarea paginii să utilizăm standarde legate de XML, cum ar fi foi de stil în cascadă (CSS – Cascading Style Sheets) sau limbajul extensibil al foilor de stil (XSL – Extensible Stylesheet Language). De asemenea, versiunea XML ne permite să păstrăm informații despre tipul datelor introduse în fiecare câmp. HTML ne permite să identificăm doar șase tipuri de date: abrevieri, acronime, adrese, citate, enumerări și variabile. Și chiar și acestea sunt cel mai frecvent utilizate (în mod greșit) pentru a modifica formatările decât pentru a identifica un câmp logic.

Limitările HTML

O dată cu dezvoltarea Web-ului din ultimii 10 ani, utilizatorii au descoperit din ce în ce mai multe moduri de a comunica între ei. Temelia acestor schimburi reciproce a fost HTML (Hypertext Markup Language) – limbajul de marcare hipertext. HTML a fost utilizat pentru a prezenta orice, de la lucrări științifice la cataloage on-line și poezii. Totuși, structura paginilor Web bazate pe HTML spune puțin despre conținutul efectiv de informații. Exemplul de definiție de listă prezentat anterior ilustrează faptul că majoritatea etichetelor oferite de HTML influențează numai macheta brută și prezentarea textului pe pagină, și chiar și informația de machetare este inflexibilă. Etichetele sunt utilizate deseori pentru a reprezenta informații în moduri care forțează sau încalcă chiar înțelesul etichetelor. Lista HTML prezentată anterior nu conține, de exemplu, definiții, ci a fost utilizată numai pentru a înșira într-un anumit mod fructe și prețuri.

Deci, au fost multe paliative în testarea și impunerea unor genuri de ordine flexibilă asupra datelor conținute în pagină. Multe dintre aceste soluții ad hoc, cum ar fi ASP sau Cold Fusion, au fost destul de reușite. Totuși, în cele mai multe cazuri ele au reprezentat abordări brevetate care trebuie reinventate pentru fiecare domeniu nou de probleme sau cer utilizarea de software și hardware al unui anumit server, care s-ar putea să nu intre în planurile dumneavoastră de achiziții.

Până acum, probabil că v-ați lovit deja de limitările HTML. Poate că ați simțit frustrarea de a nu fi capabil să descrieți *exact* ceea ce doriți să faceți utilizând structurile disponibile pentru dumneavoastră ca proiectant Web sau autor. Poate ați fost forțat să utilizați

mecanisme inaccesibile, cum ar fi cadre sau tabele, pentru a constrânge pagina dumneavoastră la o machetă tipografică nesigură, sau poate că ați utilizat etichete urâte de preformatare <pre> sau de text scris la mașină <tt> pentru a alinia corect datele. XML este un standard nou care vă permite să extindeți puterea descriptivă a documentului dumneavoastră aproape oricum și să îl modificați pentru a se potrivi diverselor scopuri. XML face posibile multe lucruri care înainte erau doar dorințe vagi.

Îmbunătățirea preciziei cu XML

XML vă permite să descrieți exact documentul dumneavoastră într-un mod care poate fi „înțeles" de un calculator. Cu toate că oamenii nu au probleme cu studierea unei pagini și deducerea semnificației unei anumite machete, cum ar fi de exemplu o factură, calculatoarele nu sunt chiar atât de inteligente. Ele au nevoie de ajutor. Etichetele XML descriptive, cum ar fi <vanzator> sau <pret> au mult mai mult înțeles pentru calculatoare decât etichetele machetă anonime furnizate curent de HTML. XML furnizează un mecanism, definiția de tip pentru document (DTD – Document Type Definition), care vă permite să partajați informațiile despre structura datelor dumneavoastră cu oricine doriți.

Validarea structurii documentului cu XML

XML vă permite să forțați validarea structurii documentului dumneavoastră. Puteți impune prezența anumitor elemente în timp ce altele le faceți opționale și să legați o structură la alta. Cu alte cuvinte, dacă doriți să includeți Element ABC, puteți obliga o instanță a Element XYZ să îl însoțească. Sau, dacă ABC este prezent, atunci XYZ poate fi exclus.

Flexibilitatea machetelor cu XML

XML face posibilă modificarea reală a prezentării documentelor în concordanță cu scopul pentru care sunt create. În loc să vă constrângă să decideți dacă un set de date este reprezentat mai bine ca o listă sau ca un tabel, îl puteți reprezenta în diferite moduri pentru scopuri diferite. Pe o pagină tipărită, un tabel este folositor iar hiperlegăturile sunt inutile, dar într-un browser audio pentru orbi, un set de liste navigabile prin hiperlegături poate fi mult mai accesibil. Introducerea acestor date într-o bază de date poate necesita normalizarea acesteia și alte transformări care ar fi inutile în alte situații. O singură sursă XML poate asigura toate aceste utilizări.

Obținerea independenței de platformă cu XML

XML este complet independent de platformă și este extrem de robust. Aceste cerințe nu pot fi satisfăcute de nici un alt mecanism de transport de date sau de procesare distribuită. XML este bazat pe text. Puteți studia datele neprelucrate și să le înțelegeți perfect. Deoarece XML descrie o simplă bază de date cu fișier plat, orice aplicație care utilizează orice tip de accesare a bazelor de date poate utiliza XML ca numitor comun pentru transportul, generarea și translatarea de la XML pentru transfer și, în același timp, poate utiliza intern formate brevetate sau normalizate. Aproape toate aplicațiile cu baze de date au deja capacitatea de a crea fișiere plate delimitate prin virgule, echivalentul înregistrărilor bazelor de date, deci extinderea virgulelor în etichete XML este aproape trivială.

Descrierea documentelor orientate spre obiecte cu XML

Cu toate că succesul pe termen lung al tehnicilor orientate spre obiecte este încă în dezbatere, XML se dezvoltă în moduri care, în cele din urmă, vor suporta ca opțiune programarea orientată spre obiecte și metodele de proiectare. Cu toate că standardul actual este puțin îmbunătățit față de modelul standard cu blocuri secvențiale, inițiative cum ar fi SOX (Schema for Object-oriented XML) – scheme pentru XML orientat spre obiecte, sau altele echivalente vor furniza acces deplin orientat spre obiecte pentru elementele gen bază de date și alte elemente ale structurilor XML.

Redefinirea modalităților de dezvoltare Web

HTML operează acum ca și un fax virtual complicat și sofisticat, care la început a fost proiectat cu intenția de a permite oamenilor de știință să împartă între ei la cerere copii ale lucrărilor științifice. Etichetele machetelor HTML reflectă structura fizică a unei astfel de lucrări, inițial o schiță simplă având asociate texte și imagini. Se etichetează anteturi, paragrafe, liste, tabele și ilustrații. Macheta prestabilită este liniară, cu textul continuând în mod obișnuit în josul paginii.

Cele câteva etichete semnificative, <CITE>, <ADDRESS> și altele, sunt destinate să scoată în evidență lucruri precum citate din articole sau cărți, o preocupare de studiu sau adresa autorului. Celelalte elemente care alcătuiesc o lucrare științifică, cartelele 3×5, calculele și clasificările, înțelegerea adâncă a structurii temei, sunt aruncate la gunoi în drumul spre hârtia artificială cu hiperlegături. Nu prea avem cu ce să lucrăm. Toate celelalte, întreaga structură a Web-ului actual, au fost îngrămădite peste acel design inițial. Este un tribut adus primilor proiectanți faptul că totul mai funcționează și chiar destul de bine.

Totuși, diferența dintre text și date este că datele au structură și context. Ființele umane pot recunoaște sau extrapola structura și contextul din indicii vizuale, deci pentru pura interacțiune umană, HTML este suficient. Dar calculatoarele nu gândesc în acest mod. Și, de fapt, nu prea gândesc.

Structura și granulozitatea datelor

XML este proiectat să permită ca orice împărțire semnificativă a unui document să fie identificată fără ambiguități ca parte a unei structuri arborescente coerente care poate fi utilizată atât de om, cât și de calculator. Deci o întreagă mașină poate fi descrisă ca o listă completă de piese, totul, începând de la motor și până la ultima piuliță, fiind desfăcut în listă de componente. Sau, o carte poate fi descrisă ca o colecție de capitole, paragrafe, note de subsol, ilustrații și toate celelalte elemente care o alcătuiesc. Este un concept uluitor, cu toate că nu poate conține încă structuri de date care nu sunt arborescente.

Multe abstractizări nu pot fi reprezentate ca arbori. Înțelesul conceptului „onoare" poate fi clar pentru un pușcaș marin, dar nu îl puteți desface în părți componente ca pe o mașină, cu toate că există concepte corelate care fac cunoscut conceptul de onoare. Deci „onoare" este fără îndoială legat de „fidelitate" într-un anumit fel, dar relația exactă dispare în atât de îndrăgita ceață a proiectanților de rețele pe care filozofii o pot extrapola în cărți interminabile.

De asemenea, Web-ul în sine nu este un arbore, ci o rețea extrem de complexă de noduri independente, legate într-un graf orientat, fără un anumit punct de pornire sau o

rădăcină. (Figurile 2.1 şi 2.2 din capitolul următor poate că vor clarifica acest lucru.) Un arbore are o singură rădăcină şi fiecare ramură este separată.

Aplicaţiile gen bază de date ale XML, cum ar fi descrierea părţilor componente ale unei maşini, sunt de obicei mai directe decât descrierea obiectelor complexe, cum ar fi o carte. Poate părea ciudat să vă gândiţi că o carte este mai complicată decât o maşină, dar este adevărat. Problema este una de *excludere*, adică asigurarea faptului că un obiect nu poate să apară, în mod necorespunzător, ca unul dintre propriii săi descendenţi. Deci o maşină are de obicei un singur motor, iar un al doilea motor nu este de obicei o parte componentă a primului motor.

Totuşi, o carte poate conţine referinţe la alte cărţi, poate să fie parte a unei serii sau poate fi o carte în mai multe volume. Aceste cărţi pot să se refere la rândul lor la cartea originală. Designul tipografic al unei cărţi poate contribui la text în moduri subtile. Structurile individuale care alcătuiesc o carte pot fi legate între ele în forme estetice complexe sau convenţionale care sunt intercalate neliniar în aşezarea tipografică. Notele de subsol şi notele de final sunt doar două exemple.

Un text medical poate conţine, de exemplu, pagini transparente suprapuse, cu scopul de a putea fi vizualizate atât individual, cât şi simultan. O carte pentru copii poate avea părţi destinate decupării şi manipulării bi- sau tridimensionale de către cititor. O carte poate conţine ilustraţii care să poată fi vizualizate cu uşurinţă numai cu un aparat stereoscopic.

Un cuprins sau un index nu constituie, strict vorbind, parte a textului liniar al unei cărţi, ci mai degrabă un instrument ataşat pentru trimiteri utile la conţinutul cărţii. Un glosar poate fi necesar pentru înţelegerea completă a textului dar rareori este citit secvenţial, sau chiar niciodată. O bibliografie sau o referinţă bibliografică pot face referiri la aproape orice tip de document sau obiect, inclusiv la filme, înregistrări audio, articole din ziare sau reviste, ca şi la cărţi, nici unul dintre acestea nefiind conţinut în text, dar putând fi ataşate cărţii.

Pe scurt, o carte în utilizarea efectivă seamănă mai mult cu un graf direcţionat decât cu un arbore, cu toate că ar fi posibilă descrierea sa secvenţială pentru a putea fi tipărită. Nu există un punct de pornire definit atunci când o citiţi; puteţi trece de la o pagină la alta nerespectând o anumită ordine şi fără să aveţi în minte un anumit scop. Este o parte a unei reţele formate din alte cărţi şi necesită o adăugire metafizică, cultura, la obiectul fizic pentru a putea fi înţeleasă. Nu puteţi crea o bază de date care să corespundă romanului *„În căutarea timpului pierdut"* de Marcel Proust fără a pierde mult la conversie.

O maşină, pe de altă parte, este în mare măsură suma părţilor şi subansamblelor sale, iar comportamentul ei este definit complet de legăturile şi componentele încorporate în ea.

Conceptul de descendenţi

Un descendent este un subansamblu sau o parte, dacă sunteţi mai obişnuit cu obiectele concrete decât cu exemplele abstracte. O maşină este formată din şasiu, motor, roţi, transmisie şi o mulţime de alte componente, toate conectate într-o ordine strictă, multe dintre ele fiind la rândul lor construite din alte piese. Un motor are un bloc, pistoane şi multe alte dispozitive cum ar fi carburatoare şi distribuitoare ataşate la acestea. Orice secţiune sau componentă a întregului ansamblu, care poate fi descrisă, este un descendent al obiectului fizic „maşină". Fiecare poate avea sau nu descendenţi. Un şurub, de exemplu, este o frunză terminală a arborelui maşină.

Pe de altă parte, într-o carte, un paragraf este posibil să nu aibă un paragraf ca descendent direct, dar *poate* avea o notă de subsol, iar acea notă de subsol poate fi atât de mare încât să aibă paragrafe în interiorul ei. Dar pot fi permise notele de subsol în interiorul paragrafelor din note de subsol? Probabil că nu. Un paragraf dintr-o notă de subsol este altceva decât un paragraf dintr-un text? XML are dificultăţi inevitabile cu excluderea, deoarece a fost eliminată special din definiţia limbajului.

XML trebuie să se bazeze pe o construcţie atentă a DTD-ului pentru a evita plasarea elementelor copil în locuri unde nu trebuie să apară, dar în anumite cazuri această strategie eşuează. Uneori trebuie să vă bazaţi pe bunul simţ al utilizatorului pentru a evita problemele.

Notă Cu toate că vă puteţi gândi la problemele de excludere ca la o slăbiciune a XML, trebuie să se ajungă la un echilibru între puterea expresivă a limbajului şi uşurinţa cu care poate fi utilizat şi implementat. Excluderea recursivă este o problemă *dificilă*. Perfecţiunea descriptivă este probabil imposibilă în toate situaţiile, din punct de vedere teoretic (Teorema lui Goedel), deci alegerea unui nivel de imperfecţiune faţă de altul este o decizie de proiectare. XML a fost proiectat pentru a fi atât util cât şi ieftin, ambele fiind criterii inginereşti solide, şi, ca orice creaţie umană, este un compromis între dorinţa pentru caracteristici abundente şi realitatea economică.

Relaţia între SGML, XML şi XHTML

XML este un *metalimbaj*, un limbaj special care vă permite să descrieţi complet o clasă de alte limbaje, care la rândul lor descriu documente. Este ca o insulă în marea SGML, un alt metalimbaj, mai puternic. XML este definit ca o „submulţime" a SGML, ceea ce înseamnă că pentru a crea XML nu se adaugă nici o componentă la SGML, dar sunt eliminate componente pentru a face limbajul mai uşor de analizat, înţeles şi utilizat. XHTML este o producţie a XML, aşa cum HTML este o producţie a SGML.

Deoarece XML este proiectat să fie *extensibil*, limbajele create cu XML sunt şi ele extensibile. Ca în SGML, descrierile limbajului create cu XML se numesc DTD-uri, definiţiile de tip pentru document (Document Type Definitions). Veţi afla mult mai multe despre ele mai târziu. Dar pentru început, aruncaţi o privire asupra SGML, izvorul XML şi al standardelor legate de acesta.

SGML

SGML (Standard Generalized Markup Language), limbajul de marcare generalizat standard, este un strămoş mai puternic al XML şi, de asemenea, un metalimbaj utilizat pentru a descrie limbaje aplicaţie. Cu toate că are multe caracteristici utile, complexitatea limbajului îl face extrem de greu de utilizat şi învăţat. Nu este pe deplin adevărat că nu a fost adăugat nimic la SGML pentru a crea XML. De exemplu, când în XML a fost nevoie de declaraţiile multiple `ATTLIST` definind acelaşi `ELEMENT`, ele au fost adăugate la SGML şi apoi reprelucrate în XML, astfel încât puteţi încă numi XML o submulţime a SGML. Proiectanţii de limbaje sunt uneori ingenioşi.

În ciuda utilizării în multe corporaţii importante şi organizaţii guvernamentale, costul şi dificultatea de învăţare şi utilizare a SGML, Hy-Time, Text Encoding Initiative (TEI) şi a altor limbaje complexe de descriere a documentelor au făcut dificilă pentru amatori şi

organizațiile mai mici accesarea imensei puteri a acestor limbaje pentru formatarea datelor structurate. XML este o încercare de a face accesibil nespecialiștilor cât mai mult din puterea SGML și a celorlalte limbaje. Un alt obiectiv este crearea simplă de implementări stabile ale unui limbaj document structurat, astfel încât costul editoarelor, validatoarelor și al altor instrumente să devină accesibil oamenilor obișnuiți.

Sugestie de la LeeAnne.com Words to weave by

Dacă proiectul dumneavoastră XML este legat de un domeniu important, interogarea firmelor și asociațiilor importante în legătură cu nevoile acelui domeniu poate dezvălui DTD-uri SGML existente. În această situație, acestea vor putea fi utilizate ca o bază pentru DTD-urile XML și vor permite proiectului dumneavoastră să se adreseze unei audiențe mai mari decât dacă ați fi pornit de la zero. Merită să cercetați în detaliu segmentul dumneavoastră de piață.

IBM, unul dintre cei mai mari editori din lume, cu documentații de sute de milioane de pagini necesare pentru calculatoarele și celelalte sisteme ale sale, a utilizat SGML din primele sale zile ca depozit de date și motor de machetare înaintea tipăririi.

Guvernul federal al SUA utilizează SGML pentru procurarea de componente, manuale, automatizarea licitațiilor pentru contractele guvernamentale, inventariere și multe alte sisteme de baze de date sofisticate. Forțele armate, programul spațial și Imprimeria Guvernului SUA utilizează SGML pentru a asigura interoperabilitatea și accesibilitatea datelor.

Deoarece XML este o submulțime a SGML, multe aplicații codate inițial în SGML au fost portate în XML. În cele mai multe cazuri, modificările necesare sunt mai mult sau mai puțin mecanice, cu toate că există unele dificultăți. Din motive tehnice, multe dintre aceste probleme vor fi rezolvate prin simplificarea descrierilor și reprezentarea lor mai exactă. Ca proiectant XML, poate vă veți trezi lucrând cu DTD-uri SGML, deci obținerea unor DTD-uri SGML inteligibile vă va ajuta în modificarea și utilizarea acestora cu succes.

Țineți minte că SGML a fost prezent de o lungă perioadă de timp, cel puțin comparativ cu durata de viață a programelor de calculator. Multe probleme au fost deja rezolvate și au instalate imense baze de utilizare. Aptitudinea de comunicare cu utilizatorii și sistemele existente poate însemna diferența dintre succes și eșec pentru începătorii care caută o piață. Cu toate că XML poate fi utilizat pentru accesarea bazelor de date SGML, necesită totuși ceva muncă de proiectare DTD.

Sugestie de la LeeAnne.com Words to weave by

Dacă segmentul dumneavoastră de piață conține agenții sau contractori guvernamentali, există probabil DTD-uri SGML care să se potrivească exact măcar cu o parte din problema lor. În acest caz există probabil grupuri din domeniu care au translatat deja acele DTD-uri la XML. Merită să vă faceți temele de casă înainte de a vă apuca de dezvoltarea unui DTD XML. Poate că este deja disponibil un DTD corespunzător.

Cu toate că nu trebuie să știți SGML pentru a învăța și utiliza XML, se întâmplă destul de des în viața de zi cu zi a unui expert XML să modifice DTD-uri SGML moștenite sau existente în DTD-uri XML. Veți învăța suficient despre acest proces pentru a-l realiza cu destulă siguranță.

Atenție O atenționare este binevenită aici. Multe DTD-uri SGML sunt imense deoarece problemele cărora li se adresează sunt mari. Nu vă puteți aștepta să înaintați grațios printr-un DTD SGML de 900 de pagini pentru dezvoltarea căruia o echipă de 20 de experți a lucrat cinci ani și să îl convertiți la XML într-o după-amiază. S-ar putea să dureze un pic mai mult. Poate un an sau chiar mai mulți.

XHTML

XHTML (Extensible Hypertext Markup Language), limbajul de marcare hipertext extensibil, este un limbaj de marcare asemănător lui HTML, definit ca un DTD XML. HTML este definit ca un DTD SGML, deci redefinirea ca o aplicație XML este mult mai utilă într-o lume XML. În particular, aceasta înseamnă că limbajul XHTML este extensibil, permițând utilizatorilor și grupurilor de utilizatori să extindă limbajul în moduri eficiente. Implicațiile acestei extensibilități cuprind:

- Documente cu o singură editare care încapsulează baze de date, manuale pentru utilizatori tipărite și funcționalități de afișare Web într-un singur fișier.
- Oportunități imense pentru tranzacții extinse de comerț electronic și automatizarea schimbului de informații utilizând vocabularele XML standardizate.
- Un mod simplu de reutilizare sau conversie a multora dintre milioanele de pagini Web HTML existente, menținând compatibilitatea cu browserele existente, dar permițând caracteristici noi pentru cei care utilizează agenți cu abilități XML.
- Posibilitatea validării automate a paginilor Web XML, eliminând complet multe erori obișnuite de codare, dar menținând compatibilitatea cu browserele moștenite.
- Furnizarea de mecanisme simple care permit Web-ului să se dezvolte pentru a satisface nevoile diverselor comunități pe care le cuprinde, fără să necesite adăugiri patentate sau nestandardizate care fac viața dificilă.

Asemenea definițiilor HTML actuale, există versiuni stricte, tranzitorii și cu cadre ale DTD-urilor XHTML. XHTML este proiectat să se adreseze unui anumit tip de probleme: cum se creează pagini Web XML care pot fi vizualizate cu browsere HTML obișnuite, fără a renunța la avantajele furnizate de XML.

Totuși, XHTML este numai unul dintre limbajele XML proiectate să se adreseze anumitor tipuri de probleme. Pe parcursul acestei cărți veți cerceta alte exemple, limbaje proiectate pentru a simplifica producerea de documente multimedia, de exemplu, sau texte cu conținut matematic.

XML în teorie și practică

XML este atât de logic încât vă veți întreba probabil de ce a durat așa de mult până să fie inventat. O parte a răspunsului este că noțiunile de bază există de multă vreme, dar au fost aplicate doar recent fișierelor de date ale calculatoarelor. O listă bazată pe componente, de exemplu, este o cerință obișnuită pentru asamblarea oricărui dispozitiv mecanic complex. Dar ideea extinderii acestui instrument de hârtie la unul electronic și generalizarea conceptului pentru a putea fi folosit la orice obiect alcătuit din părți componente, inclusiv la obiecte non-fizice, a fost o scânteiere de perspicacitate reprezentativă pentru progresul uman de-a lungul secolelor.

Oamenii organizează aproape totul în ierarhii. Este singurul mod de a aborda sarcini

într-adevăr complexe, de la achiziționarea de provizii pentru Departamentul de Apărare și până la construirea de navete spațiale. Orice structură ierarhică poate fi descrisă în XML, de la lista de componente care alcătuiesc un avion și până la structura corporației IBM. Dar XML are și unele limitări. De exemplu, nu este pe deplin orientat spre obiecte, deci utilizatorii cu seturi de probleme care necesită o abordare complet orientată spre obiecte (O-O) vor avea unele greutăți cu aplicarea XML la sarcinile lor. Totuși, există inițiative în plină desfășurare pentru extinderea domeniului XML în moduri orientate spre obiecte, deci este posibil ca această limitare să fie rezolvată în cele din urmă.

Ca un exemplu că nu sunt orientate spre obiecte, documentele XML nu pot moșteni pe deplin de la strămoșii lor și nu sunt pe deplin încapsulate, deoarece interiorul lor este complet vizibil. Ele își manifestă polimorfismul și alte comportamente orientate spre obiecte, deci au făcut un pas în direcția corectă din punctul de vedere al lumii O-O.

Între timp, există multe lucruri care pot fi făcute cu XML. Pentru început, să analizăm unele mai simple. Într-un fel, ați codat deja de multă vreme în XML. HTML este atât de apropiat de o aplicație XML încât multe pagini Web codate corect vor fi citite de un procesor XML fără probleme sau comentarii. Standardul XHTML 1.0 de la W3C reformatează DTD-urile HTML 4.0 existente, toate construcții SGML, în DTD-uri XML corecte. Aceasta va permite ca multe documente HTML existente să fie vizualizate ca documente XHTML. Deci este posibil ca paginile dumneavoastră existente să nu necesite nici un pic de muncă pentru a le transforma în XML. Multe dintre acele pagini care necesită puțină reformulare pot fi convertite în siguranță prin programe de conversie simple, cum ar fi Tidy al lui Dave Raggett de la W3C.

Notă Etichetele de marcare nu sunt singurele mecanisme care pot afecta contextul și reprezentarea textului. Există diverse mecanisme pointer utilizate pentru crearea unei baze de date de caracteristici și care indică apoi fiecare instanță a sa din documentul dumneavoastră. Microsoft Word, ca și Tagged Image Format Files (TIFF), utilizează această tehnică. Partea slabă a acestor scheme este aceea că nu sunt robuste. Dacă orice parte a antetului care conține pointerii este alterată, atunci întregul document este distrus. Unul din motivele creșterii rapide și al succesului pe Web este comportamentul robust al marcajelor bazate pe text.

Exemple practice în XML

Deci, aplicarea cunoștințelor dumneavoastră de HTML este un mod bun de a începe învățarea XML. Este un limbaj de etichete și atribute, la fel ca HTML, și utilizează multe dintre convențiile acestuia. În HTML, majoritatea conținutului este încadrat de perechi de etichete, o etichetă de deschidere și o etichetă de închidere, ca mai jos:

```
HTML: <H1>Titlu</H1>
```

XML funcționează în același mod. XML încurajează verbalismul ceva mai mult decât HTML, dar nu îl impune, deci într-o aplicație XML reală codul precedent poate arăta astfel:

```
XML: <titlu1>Titlu</titlu1>
```

În XHTML, același cod va arăta astfel:

```
XHTML: <h1>Titlu</h1>
```

care este identic, excepție făcând faptul că XHTML utilizează doar etichete scrise cu

litere mici. HTML nu este sensibil la tipul literelor, deci utilizarea convențiilor XHTML face acceptabile etichetele dumneavoastră în ambele medii.

Poate că acum este momentul să subliniem faptul că, spre deosebire de HTML, toate limbajele XML sunt sensibile la tipul literelor. Așa cum este definită eticheta, din punctul de vedere al tipului literelor, așa va trebui să o utilizați. Deci, multe aplicații XML utilizează numai litere mici și nici una nu acceptă diferite scrieri cu majuscule ale cuvintelor cheie sau ale atributelor lor.

Atributele sunt utilizate exact în același mod, excepție făcând faptul că de fiecare dată ele trebuie incluse între ghilimele. Deci, cu toate că în HTML puteți scrie

```
<IMG height=20 width=20 src="pozamea.gif">
```

în orice limbaj bazat pe XML veți scrie

```
<image height="20" width="20" source="pozamea.gif" />
```

Ați observat slash-ul din capătul drept al etichetei `<image>`? Acesta subliniază o altă diferență a XML. Chiar dacă HTML permite ca anumite etichete să fie utilizate fără etichetele de închidere, XML nu permite acest lucru. Fiecare etichetă trebuie închisă, chiar și atunci când este o etichetă vidă, ca eticheta HTML `<IMG>` sau eticheta XHTML `<img>`. Cea mai bună modalitate de a realiza acest lucru este de a plasa un slash la sfârșitul etichetei, separat printr-un spațiu de restul atributelor și conținutului. Acest lucru nu pare să întrerupă nici un browser HTML și îi spune lui XML că eticheta este închisă.

```
exemple compatibile XML/HTML: <br /> <img src=" ... />
```

Această sintaxă poate părea ciudată la început, dar dacă vă gândiți la ea, începe să aibă sens. Informația DTD HTML, sau ce transferă la majoritatea browserelor, este conținută intern chiar de către browser. XML permite ca un analizor să o valideze sau nu, caz în care nu poate accesa DTD-ul pentru a diferenția etichetele cu conținut de etichetele vide. Deci, trebuie să comunicați direct cu analizorul utilizând acest format simplu.

Ordinea etichetelor XML

XML nu este mai îngăduitor nici din punctul de vedere al ordinii etichetelor. Etichetele trebuie închise în contextul în care au fost deschise. Întotdeauna. Deci trecerea etichetelor într-o ordine greșită, permisă de majoritatea browserelor HTML în ciuda faptului că oficial nu este permisă, este interzisă în XML. Prima etichetă deschisă este întotdeauna închisă ultima:

Greșit: `<i><b>Italics Bold Text</i></b>`

Corect: `<i><b>Italics Bold Text</b></i>`

Sugestie de la LeeAnne.com Words to weave by

Ordinea etichetelor este importantă! *Context* este un alt termen pentru domeniu de valabilitate, deci dacă asemănați etichetele cu farfuriile dintr-o stivă de farfurii dintr-un bufet cu autoservire, atunci nu greșiți. Dacă deschideți o etichetă `<i>` în interiorul domeniului de valabilitate sau al contextului unei etichete `<b>`, ea trebuie închisă în interiorul aceluiași domeniu de valabilitate sau context. Nu puteți să o faceți după închiderea etichetei `<b>`. Ea trebuie să fie închisă înainte.

Distanța dintre etichete nu contează. Contextul etichetei se comportă ca o stivă de farfurii dintr-un restaurant cu autoservire. Etichetele trebuie să iasă din stivă în ordinea

inversă a intrării lor în stivă. Cu alte cuvinte, nu puteți traversa arborele structurii documentului sărind din creangă în creangă asemenea unei maimuțe. Trebuie să îl parcurgeți asemenea unei furnici.

Sugestie de la LeeAnne.com Words to weave by

Primul intrat, ultimul ieşit. Această regulă simplă este uşor de urmat, iar dacă sursa este tipărită frumos, este şi foarte vizibilă. Fiecare nivel de imbricare trebuie să îşi închidă toate etichetele înainte de ieşire. Şi toate aceste metafore sunt insinuate de structura arborescentă a documentelor XML.

Vorbind de arbori, majoritatea lucrurilor pe care le utilizați zilnic au părți componente mai mult sau mai puțin invariante, care pot fi organizate într-o ierarhie arborescentă. Un flacon de aspirină, de exemplu, are ca părți componente un recipient, un capac sau un dop de un anumit tip, una sau mai multe etichete care descriu conținutul și conținutul însuși. Dacă vi s-a repartizat sarcina de a ambala flacoane cu aspirină, veți observa că pentru a evita risipirea tabletelor există o ordine de efectuare a acțiunilor. Prima dată se lipesc etichetele, apoi se pun tabletele și apoi capacul. Ordinea capac, etichetă, tablete nu va funcționa deloc. Ordinea tablete, etichetă, capac *poate* că va funcționa, dar în cazul răsturnării flaconului riscați să risipiți tabletele. Deci, o descriere XML poate fi cea din Listingul 1.1.

Listingul 1.1 Un flacon de aspirină desfăcut pe componente

```
<bottle>
  <top>
    type 3 childsafe
  </top>
  <body>
    <body-type>
      100 count plastic
    </body-type>
    <contents>
      <count>
        100
      </count>
      <content-type>
        aspirin
      </content-type>
    </contents>
  </body>
  <labeling>
    <frontlabel>
      XYZ brand generic
    </frontlabel>
    <rearlabel>
      XYZ directions and warning
    </rearlabel>
  </labeling>
</bottle>
```


Fiecare subelement este imbricat corect în interiorul elementului care îl conține și există exact un singur element rădăcină, flaconul. Aceasta este cerința principală pentru ca un fișier XML să fie considerat construit corect. Există desigur multe moduri de descriere a unui flacon cu aspirină, iar fiecare descriere este mai mult sau mai puțin arbitrară. De exemplu, un anumit flacon poate fi introdus într-o cutie și poate avea un text tipărit. Sau, pentru anumite piețe, tabletele pot fi ambalate într-o folie. Sau, flaconul poate avea mai multe etichete în limbi diferite. Posibilitățile sunt nesfârșite, ca și lumea mărfurilor și serviciilor. Oricum, flaconul ca întreg este format dintr-un recipient, etichete, capac și aspirină. Un lucru care poate fi înțeles ușor este că tabletele de aspirină sunt conținute în flacon. Mai puțin evident poate fi faptul că flaconul în sine este alcătuit din componente care sunt imbricate din punct de vedere logic în conceptul de „flacon" utilizat aici. Ca o concluzie, deoarece este un recipient, un flacon de aspirină este cel mai intuitiv și mai accesibil exemplu de imbricare.

Sugestie de la LeeAnne.com Words to weave by

Dacă flacoanele de aspirină nu sunt destul de sugestive, încercați să vă gândiți la orice alt obiect sau la o idee care are o ierarhie. Oamenii fac tot timpul astfel de clasificări, de la organigrame de firme (fără linii punctate) până la clasificări ştiinţifice ale plantelor sau animalelor. Orice lucru care conţine o listă de componente este o ierarhie. Păpuşile Matrioşca sau un set de castroane de diferite dimensiuni sunt de asemenea exemple sugestive.

Taxonomia este știința dificilă a împărțirii lumii analogice în clasificări digitale. Diverse grupuri de firme s-au adunat pentru a hotărî cum să creeze nume și structuri standard pentru produsele lor și pentru componente. În acest mod, un fabricant de flacoane îi poate spune unui producător de aspirină ce tipuri de flacoane sunt disponibile în catalog și cât costă acestea. Sau, un producător de aspirină poate să comunice unei farmacii ce tipuri de ambalaje sunt disponibile. Instrumentele prietenoase cu utilizatorul vor ușura crearea de documente descrise în acest mod și vizibile în Figura 1.1.

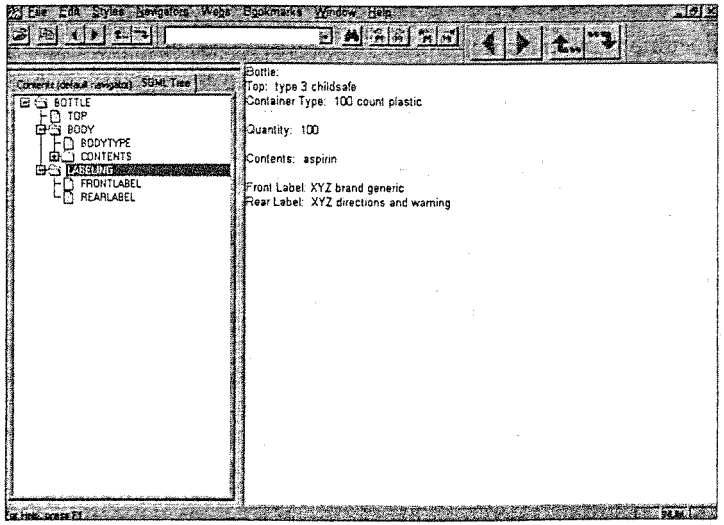


FIGURA 1.1
O imagine a Multidoc Pro Publisher editând documentul XML al presupusului nostru flacon de aspirină, având afişată structura de date XML.

Definiţiile tipurilor documentului

Un DTD (*Document Type Definition – definiţia tipurilor documentului*) XML este la un anumit nivel doar o înregistrare a modului în care este utilizat XML într-un anumit document. Nu este obligatoriu să aveţi un DTD pentru a face lucruri interesante. Dar, pentru a avea acces la întreaga putere a XML, un DTD este un pas necesar.

Sugestie de la LeeAnne.com Words to weave by

Consideraţi un DTD ca fiind un plan de proiect. El reprezintă cadrul organizatoric pentru proiectele mai mari. Nu aveţi nevoie de el pentru o utilizare unică, dar dacă doriţi să partajaţi informaţii cu oricine altcineva, sau dacă proiectul este mare, ar trebui să aveţi unul.

Fără un DTD, nu puteţi utiliza cu uşurinţă seturi de caractere extinse sau codificări. Şi nu puteţi folosi nici cele mai puternice instrumente de creaţie, care validează şi impun structura adecvată pentru document, pe măsură ce îl scrieţi. Entităţile parametru, o puternică facilitate macro care uşurează viaţa dezvoltatorilor XML, sunt şi ele inaccesibile. În cele din urmă, va trebui să furnizaţi valori pentru fiecare atribut, deoarece pentru a seta valori prestabilite este nevoie de un DTD.

Cu toate aceste motive pentru a avea un DTD, singurul motiv pentru a nu avea unul este dificultatea de a-l crea, care, după cum se ştie, este o sarcină încâlcită pentru sistemele complexe de documente. Dar nu trebuie să fie *chiar* atât de dificilă. XML a fost proiectat pentru a fi uşor de utilizat şi de înţeles. Trebuie învăţate doar câteva cuvinte cheie, iar structura limbajului este simplă. Nu este ştiinţa rachetelor. DTD-urile sunt discutate mai amănunţit în Capitolul 3, „Definiţiile tipurilor documentului în XML".

În această carte veţi învăţa cum să modificaţi un DTD existent şi chiar cum să construiţi unul pornind de la zero, sau cum să transformaţi un DTD SGML simplu într-unul XML. După ce aveţi un DTD care funcţionează, puteţi utiliza instrumente de programare pentru a manipula automat datele din documentele dumneavoastră în aproape orice mod la care vă puteţi gândi. Veţi studia aplicaţii care rescriu codul dumneavoastră XML în HTML obişnuit, pentru ca orice browser să îl poată vedea în modul dorit de dumneavoastră. Veţi face experimente analizând structura propriului dumneavoastră document pentru a demonstra cât de uşor pot fi accesate datele dumneavoastră cu Java, Perl, JavaScript şi alte limbaje de programare. Ce veţi face cu această aptitudine extraordinară depinde numai de dumneavoastră.

Dezvoltări viitoare

XML este un standard în mişcare, deşi proiectanţii de la W3C au divizat cu înţelepciune, în multe părţi componente, întregul care ar putea fi numit XML. Aceasta garantează că în loc să se aştepte să se cadă de acord asupra întregului, părţile mai mici asupra cărora diversele grupuri de lucru W3C au ajuns la un acord, pot fi emise ca Recomandări W3C. Chiar şi în acest moment se fac comentarii finale la o diversitate de extensii ale Recomandării XML fundamentale, iar pentru altele au loc dezbateri energice. Poate că, o vreme, vor coexista mai multe soluţii concurente, aşa cum se întâmplă deseori în discordantul şi cvasi-anarhicul mediu al Web-ului. Furnizorii comerciali propun soluţii pe care le-au aplicat deja, sperând să fructifice faptul că sunt primii pe piaţă, cu toate că grupuri de utilizatori propun altele, pe care le văd mai mari, mai bune, mai rapide sau mai altfel. Este un adevărat circ.

XLink şi XPointer

Printre cele mai interesante dezvoltări există două propuneri înrudite, prezentate pentru prima dată ca limbajul de legare XML (XLL – XML Linking Language), dar desfăcute acum în componente mai mici, numite XLink şi XPointer.

XLink şi XPointer rezolvă o problemă importantă a Web-ului. Hiperlegăturile HTML actuale sunt strict unidirecţionale, având o selectivitate redusă. Obţinerea unei granulozităţi mai bune în legarea la un document necesită acces la scriere în codul său sursă. Puţin probabil că acest lucru este satisfăcător într-un mediu dinamic şi mai puţin decât benign cum este Web-ul, unde accesul este apărat cu grijă împotriva răufăcătorilor şi a idioţilor. Ca exemplu, într-un document HTML obişnuit, o hiperlegătură poate conduce numai la începutul documentului sau numai la locaţii cu nume din acel document, unde este posibil să existe o etichetă ca aceasta:

```
<a name="unNume"></a>
```

Dacă acea etichetă nu există şi dacă nu puteţi convinge autorul documentului să o plaseze unde doriţi dumneavoastră, atunci nu vă puteţi lega la ea utilizând HTML. Dacă eticheta există, vă puteţi lega uşor la ea astfel:

```
<a href="documentURL#unNume">Link</a>
```

Dacă nu, trebuie să găsiţi pe cineva care are acces la scriere în document sau să vă descurcaţi fără hiperlegătură.

Atenţie Dependenţa de stilul HTML de acces la scriere, chiar în cadrul companiei dumneavoastră, este rareori înţeleaptă, cu toate că pentru setul de instrumente actuale este nevoie de el. Există multe departamente care nu vor fi fericite să dea acces la scriere utilizatorilor externi, exemplele cele mai obişnuite fiind departamentele de resurse umane, salarizare sau contabilitate, şi, având în vedere că tot mai multe aplicaţii de departament dispun de suport Web, întreţinerea unui sit care are înglobate legături va deveni un coşmar. Dar cu XLink şi XPointer, documentele pot fi accesate fără a ţine cont de locaţii cu nume existente, cu toate că acest mecanism este în continuare disponibil.

XLink extinde această aptitudine elementară cu aptitudini complexe de *transcludere*, sau de includere în documente gazdă a unor copii de la distanţă a unor porţiuni din diverse documente, de modificare a prezentării sau a destinaţiei legăturilor în funcţie de criterii definite de utilizator şi de a face alte lucruri distractive cu date distribuite. Prin adăugarea XPointer, puteţi accesa chiar modelul obiect al documentului pentru a permite ca într-un document la distanţă să fie definite oriunde ţinte virtuale. În viitorul apropiat, este posibil să se implementeze redarea dinamică a paginilor personalizate bazate numai pe criterii de legare. Se ajunge la punctul în care nu vor exista două persoane care să vadă aceeaşi pagină Web fără să aleagă în mod deliberat acest lucru.

XSL – foi de stil cu steroizi

XSL (Extensible Stylesheet Language – limbajul extensibil al foilor de stil) este o altă iniţiativă pe cale să dea roade. Recomandările CSS (Cascading Style Sheet – foaie de stil în cascadă) şi CSS2 actuale se mărginesc la retuşuri cosmetice ale aspectului unui document existent, bazat pe etichete şi atribute HTML sau XML. Modificarea radicală a unui document se putea face doar utilizând Dynamic HTML, cu problemele de compatibilitate şi durerile de cap aferente. Dar XSL permite, prin utilizarea XPath, accesul de

scriere și rescriere la întregul document, o descriere standard a structurii documentului, inclusiv părți transcluse aduse de XLink și bazate pe întregul model obiect al documentului. Veți studia în detaliu XLink și XPointer în Capitolul 9, „XLink și XPointer".

Puteți, de exemplu, să codați o foaie de stil XSL pentru a genera automat un cuprins prin traversarea arborelui documentului, trecând cuprinsul generat ca un prefix al documentului și apoi afișând documentul însuși. Sau puteți rearanja pagina în sine, ștergând știrile sportive și introducând în locul lor rezumate despre noi descoperiri astronomice, sau eliminând știrile naționale în favoarea celor de interes local. Două dintre cele trei subcomponente ale XSL, XSL Transformations și XPath, au fost deja publicate ca Recomandări. Ultima piesă este XSL propriu-zis, limbajul de formatare. Atenție, DHTML! Iar pentru majoritatea script-urilor, „La revedere"! Aceste lucruri le veți studia mai îndeaproape în Capitolul 12, „CSS1, CSS2, DSSSL, XSL".

Una peste alta, o mulțime din lucrurile pe care acum, în locul dumneavoastră, le realizează portalurile Internet, vor fi posibile acasă. Furnizorii vor vinde sau vor oferi foi de stil care pot fi personalizate pentru ca dumneavoastră să puteți accesa informațiile care vă interesează și să aruncați la gunoi intervențiile aleatoare nedorite.

Semnăturile XML vor permite ca paginile Web să fie semnate de către autor într-o manieră care nu va putea fi negată, permițând încorporarea unor măsuri de securitate într-o rețea ale cărei informații par uneori cât se poate de suspecte.

O descriere concisă a originilor XML

Marcajele au fost alături de noi aproape de la apariția limbajului scris. Egiptenii, care au folosit unul dintre cele mai vechi sisteme de scriere din lume – hieroglifele –, marcau numele de persoane cu un cartuș oval pentru a le distinge de cuvintele obișnuite. Utilizau, de asemenea, culori pentru a evidenția frazele importante, un fel de aldine egiptene. Deoarece egiptenii puteau scrie, în funcție de context, fie de la stânga la dreapta fie de la dreapta la stânga, ei prezentau imaginile animalelor sau oamenilor cu fața spre începutul liniei, ca un atribut sau marcaj bidirecțional antic. Ei considerau că este mai politicos ca, pe măsură ce citiți un rând, să fiți întâmpinat de personaje. Ar fi fost nepoliticos dacă personajele ar fi stat cu spatele spre dumneavoastră. Figura 1.2 prezintă o porțiune de pe Piatra Rosetta, una dintre cele mai importante pietre de hotar în înțelegerea comunicării și a istoriei.

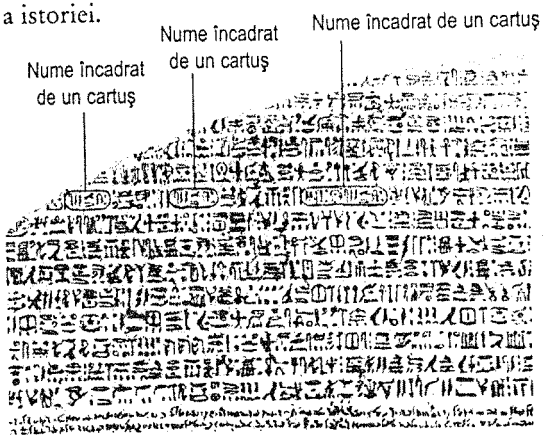


FIGURA 1.2
Hieroglife egiptene, datând aproximativ din 196 îC, exemplificând utilizarea cartușelor de marcare pentru nume și a marcajelor bidirecționale.

Convenții de marcare în ebraica veche

Ebraica veche, în care a fost scrisă Tora, nu conține aproape deloc marcaje, excepție făcând doar „coroanele" decorative așezate peste șapte litere și două forme de litere finale. Asemenea multor sisteme de scriere, în ebraica veche a apărut târziu ideea de a pune spații între cuvinte, una dintre primele scheme de marcare, dar acum Tora conține aceste invenții moderne.

Notă Cu toate că aceste sugestii referitoare la limbaj sunt concise, este important să rețineți că XML este proiectat să suporte toate limbile omenești. Deci, ebraica este doar un exemplu simplu pentru o clasă de probleme cu care se poate confrunta un proiectant XML.

Un fragment din Tora (tradus din ebraică) ar putea arăta astfel:

ACEASTA ESTE O PROPOZITIE SCRISA IN STILUL TORA ACEASTA ESTE PROPOZITIA URMATOARE ACEASTA ESTE CEA DE A TREIA PROPOZITIE SI ASA MAI DEPARTE

Doar că ebraica veche nu folosea vocale, preferându-se înlocuirea lor cu un proces de gândire, deci propozițiile de mai sus ar arăta de fapt astfel:

CST ST PRPZT SCRS N STLL TRH CST ST PRPZT RMTR CST ST C D TR PRPZT S S M DPRT

Citirea acestui text este întrucâtva o sarcină descurajantă, chiar dacă dumneavoastră cunoașteți cuvintele pe de rost. Textul este citit de obicei cu ajutorul unui indicator pentru a nu vă pierde în cadrul paginii.

Nu uitați, marcajul introdus de o persoană este un ajutor esențial pentru ca o altă persoană să înțeleagă și ea. Este destul de obișnuit să se facă versiuni speciale pentru cititorii Tora, cu vocale marcate și indicații de intonație pentru citirea psalmului, pentru a-i ajuta pe cei cu memoria slabă sau mai puțin talentați.

Convenții de marcare în latina veche

Istoria propriei noastre scrieri latine este similară, cu spațierea cuvintelor, cu punctuație retorică, cu introducerea literelor mici pentru a face o distincție mai clară între cuvintele obișnuite și cele importante și alte inovații asemănătoare, destinate să facă scrierea mai ușor de înțeles și de utilizat. În timpul Renașterii a fost introdus caracterul punct și virgulă pentru a marca o anumită relație subtilă între idei, dar ne-am chinuit destul de mult timp până acum folosind acest sistem de marcare vechi de sute sau chiar mii de ani.

Îl utilizăm zilnic, subliniind cuvinte, folosind semne de punctuație în fraze, scriind cu majusculă numele proprii și primul caracter al fiecărei propoziții pentru a face mai clare pe hârtie ideile noastre.

Totuși, sistemul nu a fost suficient de bun pentru formatarea cărților tipărite. Tipografii au dezvoltat un sistem de marcare inteligent pentru a identifica și a aplica modificări de formatare la părți ale unor cărți sau ale altor tipărituri. Marcajele tipografilor se numesc chiar marcaje, iar numele provine de la faptul că ele au fost adăugate manual, „marcate" pe paginile dactilografiate sau pe manuscrise. Toate trucurile din domeniul tipografic includ marcaje speciale cum sunt aldine, cursive, majuscule reduse, indentări și diferite tipuri de spații, schimbări de fonturi, tratamente speciale pentru titluri, titluri de capi-

tole, subsol, antet, casete, numere de pagină și multe altele. Figura 1.3 prezintă o pagină dactilografiată, marcată de un editor, cu mențiuni referitoare la stilul tipografic și prelucrarea textului.

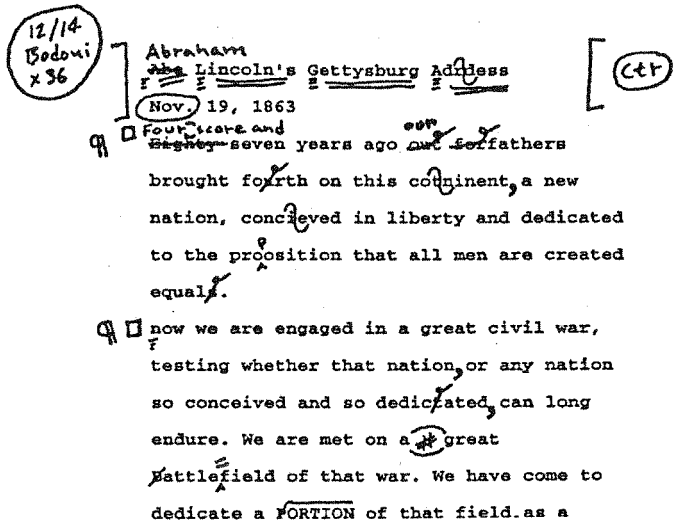


FIGURA 1.3
Un exemplu de document dactilografiat, marcat cu notațiile editorului adresate tehnoredactorului.

Marcajele în istoria recentă

O dată cu începutul erei calculatoarelor era nevoie de un model nou care să permită realizarea de hârtii și documentații formatate. Unul dintre primele, Runoff, a fost codat de Jerry Saltzer pentru sistemul de operare CTSS pe IBM 7094 la începutul anilor '60 și portat apoi pe Multics, un alt sistem de operare timpuriu. Runoff a fost primul instrument important portat pe noul inventatul sistem de operare UNIX de la laboratoarele Bell. Runoff și toți descendenții săi utilizau etichete de marcare speciale pentru a controla aspectul formatat al textului, așa cum facem noi azi. Din multe puncte de vedere poate fi considerat strămoșul lui HTML și al XML, cu toate că aplicațiile moderne au depășit cu mult scopul pentru care au fost proiectate aceste programe incipiente.

Un canadian, William Tunnicliffe, a cerut în 1967 separarea în media electronică a informațiilor de formatare de informațiile de conținut. Aproape simultan, Stanley Rice, un redactor din New York, a propus un sistem de etichete universale cu structură editorială. O idee excelentă se năștea și lua avânt.

Pe baza acestor idei a apărut conceptul de codare generală (Generic Coding), iar din acesta GML (Generalized Markup Language – limbajul de marcare generalizat) creat în 1969 de Charles Goldfarb, Edward Mosher și Raymond Lorie de la IBM pentru codificarea și publicarea cu succes a milioane de pagini de documentație IBM.

După o perioadă de dezvoltare, GML și Generic Coding s-au reunit din nou în 1978 sub forma unui comitet pentru dezvoltarea unui nou limbaj, bazat pe GML dar extins cu idei noi, venite de la mulți oameni. În 1980, SGML (Standard Generalized Markup Language – limbajul de marcare generalizat standard) a fost publicat ca schiță. În 1983, noua schiță de standard a fost publicată ca un standard în domeniu, GCA 101-1983, și adoptat de Fisc și de Armata SUA.

Între 1983 și 1987 s-a lucrat la un standard de publicare SGML pentru cărți, publicații periodice și articole cu componente opționale din matematică, lucrări științifice și tabele complexe. El a fost adoptat ca standard ANSI, Z39.59, în 1988 și, mai târziu, într-o formă revizuită, ca standard internațional, ISO 12083.

În 1986, standardul internațional SGML, ISO 8879:1986, a fost publicat iar lucrurile au început să se precipite. În 1988, Departamentul de Apărare al SUA a publicat standardul SGML CALS (Computer-aided Acquisition and Logistical Support – suportul logistic și de achiziție asistat de calculator) ca MIL-M-28001.

De atunci, multe proiecte mari bazate pe SGML au înflorit într-o relativă obscuritate pentru cei din afara cercurilor științifice și din domeniu – Text Encoding Initiative (TEI – inițiativa de codare a textului), HyTime, ISO 10744, Hypermedia Time-based Structuring Language, un standard foarte complex, cu suport multimedia sincronizat de toate tipurile, și multe altele.

Toate aceste proiecte au fost imense, necesitând mult studiu pentru familiarizarea cu limbajul și pentru ca facilitățile să devină eficiente în marea lor diversitate de posibilități. Și iată că toate acestea ne-au condus la XML.

Sintaxa XML

CAPITOLUL

2

În acest capitol

Structura XML
Descrierea de vocabulare noi cu XML
XML, doar un alt HTML?
Startul în XML
Definirea unui document XML ca întreg
Constrângerile construirii corecte
Prologul: Declarația XML
Construirea prologului unui document XML: Declarația tipului documentului (Document Type Declaration)
Crearea corpului documentului
Formarea structurilor logice în XML
Studii de caz

Structura XML

XML este format de fapt din două limbaje, de fapt metalimbaje, ambele descrise în același document. Primul este un set de reguli pentru realizarea de documente XML construite corect, în timp ce al doilea este un set de reguli pentru realizarea unei *definiții a tipului documentului* XML, sau DTD (Document Type Definition), care permite ca structura documentului XML să se supună unor constrângeri și să fie validată față de acele constrângeri. Distincția dintre aceste două limbaje este deseori neclară, deoarece un document XML complet presupune măcar existența opțională a unui DTD, indiferent dacă acesta este de fapt prezent sau nu. Pentru a complica și mai mult lucrurile, un DTD poate fi compus din două părți, o *submulțime internă* și o *submulțime externă*.

Acest capitol studiază documentul XML fără a insista prea mult asupra DTD-ului, deoarece un document XML poate fi creat și fără a face referiri la un DTD. Din motive de performanță, multe documente XML vor fi utilizate fără a le valida în raport cu DTD-ul, chiar dacă DTD-ul este disponibil. În cazul conexiunilor lente, citirea dintr-un DTD nesituat pe sistemul dumneavoastră poate fi îngrozitor de înceată, iar datorită faptului că un DTD poate conține referințe la alte documente, rezolvarea tuturor referințelor externe poate dura exagerat de mult chiar și în cazul conexiunilor de mare viteză. Utilizatorii sunt obișnuiți ca documentele HTML să fie încărcate incremental, deci pot citi înainte ca documentul să fie încărcat în totalitate, dar analizoarele XML de

validare nu permit afișarea documentului decât în cazul în care acesta este valid, deci documentul va apărea pe ecranul utilizatorului numai în momentul în care este încărcat în totalitate. Acest lucru poate fi supărător.

Totuși, fiecare document este creat având în minte un DTD, indiferent dacă DTD-ul este explicit sau nu. Chiar și la crearea documentelor fără un DTD, trebuie să aveți în minte un fel de DTD empiric pe măsură ce creați documentul, deoarece un DTD descrie o structură de date.

Detalii tehnice despre XML

Anexa A, „Referințe XML/XHTML", descrie (parțial) sintaxa fundamentală XML prin listarea notației Extended Backus-Naur Form (EBNF), forma Backus-Naur extinsă, utilizată chiar de standardul W3C XML 1.0 pentru a o defini. Regulile de producție EBNF selectate sunt completate cu descrieri concise ale constrângerilor explicite de validitate și/sau de construire corectă cu care W3C a modificat sau a extins anumite reguli EBNF. Cu toate că forma EBNF este nemaipomenit de compactă, permițând ca producțiile legale într-un limbaj de programare să fie definite succint ca o serie de reguli de transformări gramaticale, pentru simplitate, W3C a lăsat pe dinafară destul de mult din descrierea EBNF a limbajului. Majoritatea părților care lipsesc se referă la constrângeri și la descrieri verbale ale criteriilor adiționale pe care o anumită producție trebuie să le respecte, dar unele se află de fapt în textul standardului.

EBNF este doar o metodă prescurtată de a spune lucruri despre limbaje de programare, iar dumneavoastră puteți descrie la fel de ușor XML într-un mod mai accesibil, utilizând cuvinte, cu toate că EBNF este mai concis în comparație cu echivalentul din engleza obișnuită. Totuși, unele lucruri sunt dificil (sau obositor) de exprimat în EBNF, iar EBNF în sine este concis până la limita obscurității pentru majoritatea oamenilor, așa că W3C a hotărât să combine în recomandarea sa cele două tipuri de descriere a limbajului. Unele reguli de producție sunt descrise complet în notația EBNF, iar textul însoțitor este doar o explicație a ceea ce spune de fapt EBNF. Unele reguli de producție sunt descrise complet numai printr-o combinație a regulii EBNF și a comentariului aferent din text, deci regula EBNF servește ca un schelet general la care trebuie adăugate textele cu comentarii pentru a caracteriza regula efectivă.

În realitate, EBNF-ul Recomandării XML 1.0 a W3C este și trebuie să fie suplimentat cu diverse constrângeri textuale, conținute în corpul documentului, pentru a-l înțelege cu adevărat. Dar după ce ați înțeles cum se potrivesc toate acestea, regulile EBNF sunt utile ca fițuici care să vă reamintească modul de funcționare al limbajului.

Notă Cu toate că există o sintaxă EBNF pentru XML, acea descriere nu este completă. Fără a lua în considerare numeroasele constrângeri trecute în Recomandare și fără citirea atentă a textului, înțelegerea greșită și implementarea incorectă sunt aproape inevitabile.

Acesta este motivul pentru care listingul EBNF vă stă la dispoziție în anexă, nu pentru a învăța cum să folosiți XML, ci pentru a vă reaminti cum trebuie utilizat atunci când căutați o anumită construcție și nu doriți să răsfoiți întreaga specificație. Deoarece Recomandarea XML 1.0 este extrem de concisă, poate veți dori să puneți un semn de carte la Recomandarea XML 1.0 adnotată de Tim Bray (Tim Bray's Annotated XML 1.0 Recommendation), la `http://www.xml.com/axml/testaxml.htm`, deoarece explică mai clar

unele dintre concepțiile care au intrat în specificație. Rețineți, de asemenea, că trebuie să citiți cu atenție și erata specificației. Au fost făcute multe modificări importante și, cu toate că aceste modificări se regăsesc în această carte, este posibil ca în viitor să apară și altele.

Poate că toate acestea sună complicat, dar de fapt nu este așa. Sunteți probabil obișnuit cu respectarea unor reguli care se pot modifica în funcție de context. De exemplu, regula care spune că trebuie să opriți la lumina roșie a semaforului atunci când conduceți o mașină este o regulă simplă și universal valabilă, deci atunci când învățați să conduceți o rețineți rapid. Dar, în unele state, o modificare specială a regulii luminii roșii vă permite să virați la dreapta după ce ați oprit, dacă nu cumva se specifică altceva. Și aceasta pare o regulă simplă, dar ar trebui să aibă atașată o constrângere de validitate deoarece nu în toate statele este permis acest lucru. O notă de subsol ar fi suficientă în majoritatea cazurilor. O alternativă ar fi crearea altor reguli similare referitoare la lumina roșie, având încorporate condiții speciale pentru tratarea excepțiilor. Astfel de lucruri sunt dificil de făcut în EBNF, așa că W3C a rămas la o regulă simplă și a adăugat constrângeri, care sunt mai mult sau mai puțin ca niște note de subsol pentru regulile care au nevoie de ele.

Descrierea de vocabulare noi cu XML

XML are o natură dual: un metalimbaj care vă permite să descrieți noi structuri de documente și vocabulare și un limbaj utilizat ca să exprime acea structură și acel vocabular în cazul unui document. Există o diferență clară între un document XML, care poate avea sau nu asociat un DTD exprimat în metalimbajul XML, și un DTD XML. Ele utilizează sintaxe complet diferite pentru a descrie un document XML, unul prin exemplu iar celălalt prescriptiv.

Definițiile tipului documentului XML (DTD-urile) descriu instanțe ale limbajului XML, numite uneori vocabulare XML. Documentele XML sunt create utilizând acele limbaje. Din păcate, această diferență se pierde uneori în exprimarea neoficială, iar vocabularele XML particulare și DTD-urile asociate sunt descrise inexact ca „XML".

Cu toate că trebuie să le cunoașteți pe ambele pentru a stăpâni pe deplin XML, nu este necesar să definiți un DTD pentru a crea și utiliza un vocabular XML, atât timp cât respectați regulile. Utilizatorul unui limbaj sau vocabular XML poate că nu va vedea niciodată sau poate că nici nu îi pasă de DTD-ul utilizat pentru a descrie aplicația sa, la fel cum miilor de persoane care lucrează în design Web utilizând HTML nu le pasă de W3C HTML 4.0 SGML DTD utilizat în descrierea HTML. De fapt, un DTD poate să nu existe. Nu are o importanță chiar așa mare la nivelul aplicației. Totuși, pentru că citiți această carte, se presupune că veți fi chemat să proiectați sau să lucrați cu DTD-uri, iar pentru o înțelegere adâncă a XML trebuie să înțelegeți cum și de ce sunt construite și utilizate DTD-urile.

Avantajele definiției tipurilor documentului

Cu toate că DTD-urile sunt opționale deoarece un procesor XML poate deduce un DTD convenabil dintr-o instanță a XML, existența unui DTD oferă multe avantaje:

- Un DTD descrie organizarea unui document într-un mod care poate fi distribuit cu ușurință.
- Un DTD permite unui proiectant să creeze o transformare *robustă* dintr-un anumit tip de document XML într-un alt format, pentru afișare sau transfer. Deoarece cu un DTD știți absolut orice despre documente, veți ști cum să tratați structuri care este posibil să nu existe într-un anumit exemplu, dar sunt permise de tipul documentului, chiar dacă nu le-ați întâlnit niciodată până acum.
- Un DTD permite unui analizor care *validează* să determine dacă un anumit document este sau nu creat în conformitate cu regulile stabilite de autorul specificației. Acest lucru este extrem de important pentru EDI (Electronic Data Interchange – schimb electronic de date) și alte aplicații în care documentele vor fi partajate și utilizate și de alte procese.
- Fără un DTD, un mediu de creare XML nu poate oferi sugestii despre elementele care sunt necesare sau opționale într-un anumit loc și despre atributele pe care le poate conține un anumit element. Meniurile contextuale sau sugestiile pot fi de mare ajutor în accelerarea dezvoltării documentelor și în prevenirea erorilor.
- Fără un DTD, autorul unui manual de creație sau al unui document de stil nu are de unde să știe cum ar trebui construit documentul definit. Un manual de creație este o întrupare a informației exprimate într-un DTD, cu toate că nu este un DTD în sine.
- Specificarea DTD-ului utilizat într-un document identifică versiunea standardului folosit la crearea sa. Când documentele se dezvoltă în funcționalitate și sintaxă, acesta poate fi un indiciu important referitor la afișarea sau transformarea unui document într-o situație nouă.

Disponibilitatea unui DTD oferă informații și beneficii semnificative *dacă* aveți nevoie de astfel de beneficii. Dar asemenea oricărui lucru din viață, acesta atrage după sine și un anumit cost.

Combaterea dezavantajelor definiției tipurilor documentului

Cu toate avantajele lor, DTD-urile ridică și probleme. Ele utilizează o sintaxă diferită de XML, deci construirea unui DTD necesită un alt set de abilități. În plus, ca orice descriere tehnică, implicarea în proiectarea unui DTD fără a vă gândi la aspectul structurii datelor dumneavoastră în document vă poate împotmoli în detalii atunci când ar trebui să fiți atent la structura generală. Multe persoane proiectează documentele XML utilizând vocabularul XML destinat și apoi utilizează un instrument de extragere automată de DTD-uri pentru a genera un DTD pe baza documentului. După realizarea acestui lucru, DTD-ul poate fi reglat prin adăugări la codul sursă sau prin ajustarea acestuia.

Dar DTD-urile au și următoarele dezavantaje:

- Cu un DTD, un agent utilizator XML care validează necesită *cel puțin* încă o operație de citire pentru a accesa locația unde este disponibil DTD-ul. Cu toate că plasarea în cache poate reduce performanțele pentru unii utilizatori din rețea, multe utilizări previzibile ale documentelor XML vor face imposibilă utilizarea stocării în cache.
- Un DTD crește cu mult complexitatea analizorului necesar pentru a determina dacă un document poate fi afișat. Pentru unele dispozitive, este posibil ca acest lucru să nu fie realizabil.
- Unele medii de creare care *validează* și utilizează un DTD îngreunează salvarea spațiului dumneavoastră de lucru la sfârșitul zilei sau restaurarea sa în ziua următoare, cu

excepția cazului în care documentul se află într-o stare validă. Acest lucru poate fi supărător dacă mai aveți mult de lucru și trebuie să îl întrerupeți pentru o vreme.

- Din punct de vedere teoretic, un DTD este capabil să continue nelimitat citiri externe, deoarece un DTD poate conține alte DTD-uri sau mulțimi de entități. Este posibil ca pentru afișarea unor documente complexe să fie nevoie de foarte mult timp atunci când se utilizează un analizor care validează.

Decizia utilizării unui DTD este un compromis între abilitatea de utilizare fără restricții, obișnuită în HTML – posibilitatea de a face aproape tot ce doriți și de a drege rapid lucrurile – și un mediu mult mai structurat, în care trebuie pus punct pe fiecare „i" și linie pe fiecare „t". În multe situații, cum sunt cele în care realizați documente care trebuie să fie general disponibile și sunt create distribuit, trebuie să se respecte strict regulile și veți dori un DTD. În alte cazuri, cum ar fi atunci când dezvoltați un tip de document XML nou, nu veți avea nevoie sau nu veți dori constrângerea strictă și îl veți putea realiza fără un DTD, cel puțin în timpul proiectării inițiale.

Dar după ce dezvoltarea a devenit un produs stabil, veți dori să formați proiectul dumneavoastră astfel încât acesta să poată fi distribuit cu ușurință. Chiar dacă veți dori să creați și un manual de utilizare, un DTD este o metodă simplă de a permite utilizatorilor să își testeze propriile documente pentru a se asigura că au respectat indicațiile pe care le-au citit în manual. În acel moment, este posibil chiar să regretați că DTD-urile oferă o flexibilitate așa de mare. Dacă doriți ca un câmp să conțină un număr de telefon, definirea câmpului ca `CDATA` lasă mult de dorit.

De fapt, schemele XML oferă posibilități de a crea reguli chiar mai stricte, lucru care poate fi util în situații care cer un control foarte strict asupra conținutului unui câmp, așa cum veți vedea în Capitolul 7, „Scheme XML".

XML, doar un alt HTML?

XML este un limbaj cu etichete și atribute asemănător cu HTML, dar un HTML transformat atât de mult, încât nu mai poate fi recunoscut. XML este un HTML cu steroizi.

XML este mult mai structurat decât HTML. În timp ce procesoarele HTML acceptă în mod curent cod incorect și diform și încearcă să îi dea sens pe ecran, XML *trebuie* să se oprească atunci când găsește o eroare fatală, care poate fi aproape orice tip de eroare. Aceasta este într-un fel o întoarcere la primele zile ale prelucrării datelor, când orice eroare din cod era pedepsită cu un vidaj de memorie (core dump) lângă care puteați petrece ore întregi încercând să îl descifrați. Așteptați-vă să petreceți puțin mai mult timp depanând XML decât făceați anterior în HTML.

Totuși, pe lângă acest comportament necruțător, XML este cu mult mai puternic. În timp ce HTML se mulțumește cu vreo 77 de etichete, în funcție de cine le-a numărat, XML are un număr de etichete aproape infinit, structurate în aproape orice fel doriți.

Oricum, fundamentele au rămas aceleași, iar experiența dumneavoastră în HTML va ușura acceptarea pasului evolutiv reprezentat de XML și de standardele asociate lui. Utilizarea XML nu este atât de simplă ca rostogolirea unui bușteau, dar nu se compară nici cu escaladarea Everestului. Cu puțină disciplină și cunoștințe, pe care această carte vă va ajuta să le obțineți, veți scrie în curând cod XML.

De fapt, într-un fel ați scris cod XML încă din timpul utilizării anterioare a HTML. Nu numai că un HTML bine făcut este foarte aproape de XHTML – un înlocuitor al HTML

care respectă XML – dar un cod HTML 4.0 *curat* este destul de lizibil ca XHTML 1.0. Deoarece HTML 4.0 a fost structurat ca o aplicație SGML, iar XML este o submulțime a SGML, acest lucru este logic. Diferențele sintactice minore dintre XHTML, un vocabular XML, și HTML, un vocabular SGML, pot fi ajustate automat dacă este nevoie.

Autorul unui document XML oferă de obicei un manual de creare sau codare (sau o foaie, pentru DTD-uri mici) descriind etichetele utilizate în aplicația XML, atributele lor, valorile posibile și modul lor de imbricare. Urmărirea unui astfel de manual de codare nu este mai dificilă decât reținerea faptului că o linie de tablou `<tr>` trebuie să fie imbricată în interiorul unui tablou `<table>` și nu are, sau nu ar trebui să aibă, sens în afara acelui context.

Pentru majoritatea situațiilor, acest lucru este suficient. Autorii XML nu mai trebuie să fie niște guru ai tehnicii care să poată extrapola instantaneu structura și scopul unei aplicații aruncând doar o privire asupra DTD-ului, la fel cum proprietarul unui automobil nu trebuie să fie expert în mecanica auto. XML este capabil să ofere autorilor suficient ajutor în învățarea modului de utilizare a unei anumite aplicații, deoarece aceștia sunt încurajați să dea etichetelor nume sugestive, care să fie ușor de reținut. Autorul unei aplicații *ar trebui* să furnizeze un manual de creare care să explice în termeni simpli modul de utilizare a aplicației. Teoria este că orice viitor informatician va putea studia codul dumneavoastră XML și își va da seama ce este și cum este structurat, fără a fi nevoit să recurgă la documentația de proiectare originală (probabil pierdută în praful istoriei), bazându-se numai pe structură și numele elementelor.

Sugestie de la LeeAnne.com Words to weave by

Cu toate că orice procesor XML vă poate spune despre codul dumneavoastră dacă este sau nu construit corect, iar un manual vă poate ajuta să construiți un document valid, DTD-ul vă permite să verificați fără ambiguități munca dumneavoastră. Totuși, în funcție de tipul de instrument de creare utilizat, acesta poate fi un pas diferit de procesul de editare.

Codul dumneavoastră îndeplinește acest ideal în funcție de utilizarea pe care o dați numelor etichetelor, totuși între anumite limite:

- Numele de etichete care încep cu șirul „xml", indiferent de tipul literelor, sunt rezervate; adică, indiferent de situație, nu vă este permis să le creați. Nu le inventați după voia dumneavoastră. Dacă simțiți că *trebuie* să aveți un astfel de nume de etichetă, propuneți-l la W3C ca parte a unei „Propuneri de membru" (presupunând, evident, că sunteți un membru) și vedeți ce se întâmplă.
- Numele de etichete care conțin caracterul două puncte pot fi interpretate ca identificatori având asociată o zonă de nume, deci utilizarea celor două puncte în numele etichetelor este puternic combătută și poate fi chiar interzisă. De ce să riscați? Evitați-le.
- Un nume de etichetă trebuie să înceapă cu o „literă", care în acest context este orice literă sau ideogramă Unicode/ISO/IEC 10646, sau cu o liniuță de subliniere (sau cu două puncte, pe care știți deja că este bine să le evitați pentru a preveni confuzia cu zonele de nume).

În continuare, numele unei etichete poate conține orice „literă", ideogramă sau cifră Unicode/ISO/IEC 10646, caractere de combinare, caractere de extindere, puncte, cratime, spații sau două puncte. De altfel, puține limbi de pe glob au caractere cu care să nu poată începe un nume corect în limba respectivă. Aceste caractere sunt excluse din lista de caractere dacă se află într-o poziție în care pot fi văzute ca „primele" după o cratimă sau un alt separator logic de cuvinte. Dar în cazul în care cunoașteți limba, acest lucru va fi evident.

Caracterul thailandez *mai yamok* (*arată ca un f întors și fără liniuță*), de exemplu, arată ca o literă, dar nu poate fi folosit ca primă literă a unui cuvânt deoarece el semnifică repetarea literei anterioare.

Caracterele de combinare sunt caractere speciale, folosite pentru a accentua un alt caracter, multe dintre ele fiind normalizate la un singur caracter cu accent. Acesta este un avantaj pentru intrările de la tastatură deoarece multe limbi care conțin caractere accentuate vă permit să le introduceți utilizând caractere accent speciale „de lățime zero" care se pot atașa la orice alt caracter.

Caracterele de extindere sunt diverse semne de punctuație speciale, cum ar fi (în limbile europene) middle dot, triangular colon și half-triangular colon. Caracterele extinse sunt similare în alte scrieri din lume; nu sunt exacte din punct de vedere alfabetic dar se potrivesc pe undeva. Dacă trebuie să utilizați unul într-o altă limbă decât engleza, veți ști probabil ce este și deci va fi ușor de găsit. Dar dacă nu vorbiți sau scrieți în arabă, utilizarea unui „tatwell" arab în numele etichetei dumneavoastră este probabil un moft, cu toate că este permis.

Startul în XML

Într-o anumită măsură, știți deja cum să codați în XML dacă v-ați obișnuit să scrieți cod HTML 4 curat și corect. Poate că va trebui doar să scăpați de unele deprinderi rele pentru a deveni un autor XML priceput, așa că aici vă veți concentra asupra diferențelor dintre XML și HTML. Acest subiect evidențiază mulțimea de deprinderi necesare pentru XML și clarifică numeroasele asemănări dintre XML și HTML:

- **XML este sensibil la tipul literelor deoarece majusculele nu reprezintă un concept universal** – Dacă ați trata literele mari și mici ca fiind echivalente, ar trebui să faceți la fel pentru mii de alte variante de litere în alte limbi, o operație împovărătoare. Unele limbi nici măcar nu au tipuri de litere. De exemplu, în limba ebraică nu există litere mici, iar limba arabă face distincție între forma inițială, de mijloc și finală a literelor. Pentru cei care preferă să scrie etichetele cu majuscule și atributele cu litere mici, pentru a le evidenția, aceasta este o știre teribilă. Dar editoarele de cod moderne nu mai acordă o importanță așa de mare acestui lucru ca înainte. De exemplu, precizarea anumitor culori pentru a marca etichete este un lucru obișnuit, deci utilizarea majusculelor este întrucâtva un anacronism istoric, asemenea numerelor de linii în COBOL.
- **XML este foarte precis cu privire la imbricarea corectă a etichetelor** – Etichetele nu se pot încheia într-un context diferit de cel de început. Deci, dacă doriți `<bold><italics>`, trebuie să închideți fraza evidențiată cu `</italics></bold>`, pentru a evita o eroare fatală. Deoarece XML poate referi și include documente XML și fragmente de documente oriunde pe Web unde dumneavoastră nu aveți control, fiecare document XML trebuie să se supună acelorași reguli pentru a nu strica documentele altora.
- **XML nu este bine protejat împotriva recursivității** – Cu toate că este posibil să stabiliți excluderi explicite la un anumit nivel, la o structură complexă a unui document este dificil să mențineți acele excluderi la un nivel redus, mai ales atunci când utilizați etichete care pot fi aplicate la orice nivel. Deci, interdicția HTML referitoare la includerea unei etichete ancoră `<a>` în interiorul altei etichete ancoră există și în XHTML, dar nu este impusă dincolo de includerea directă.

- **XML vă obligă să închideți fiecare etichetă, chiar și etichetele vide** – Deoarece este posibil să creați un document XML care să nu utilizeze un DTD, un procesor XML nu are de unde să știe dacă o etichetă este sau nu vidă. Deoarece toate documentele XML trebuie să fie *construite corect*, dumneavoastră trebuie să marcați etichetele vide cu o sintaxă specială care spune unui procesor XML că eticheta este vidă și închisă. Acest lucru îl realizați plasând un spațiu și un caracter slash la sfârșitul etichetei, astfel:

  ```
  <break />
  ```

  Există o sintaxă alternativă, care este la fel de bună pentru procesoarele XML reale, dar blochează frecvent browserele Web atunci când este utilizată cu XHTML; aceasta cere ca o etichetă vidă, cum ar fi `<br>`, să fie închisă cu `</br>`, astfel:

  ```
  <br></br>
  ```

  Din păcate, este prea periculoasă pentru a fi utilizată în siguranță. Multe browsere actuale și majoritatea celor moștenite nu recunosc eticheta de închidere non-HTML și fac lucruri ciudate cu ea. Navigator 4.7, de exemplu, poate zăpăci afișarea atunci când se împiedică de o etichetă `</br>`. Comportamentul exact poate diferi în funcție de poziția în cod și de eticheta vidă care se închide. Pe scurt, această sintaxă este predispusă la erori și ar trebui evitată.
- **XML necesită încadrarea valorilor atributelor fie între apostrofuri, fie între ghilimele** – Acolo unde HTML este indulgent, mai ales în ceea ce privește numerele și aproape orice șir fără spații în interior, XML tratează totul ca șir de caractere și lasă aplicația să își dea seama singură de toate acestea.
- **XML recunoaște mai multe limbi** – Spre deosebire de HTML, seturile de caractere extinse utilizate în multe limbi europene nu sunt pe deplin recunoscute în mod prestabilit. Există un mecanism simplu pentru includerea acestora, precum și a întregului set de caractere Unicode (cunoscut, de asemenea, și ca ISO/IEC 10646) care are peste un milion de caractere, deci suportul pentru chineză, arabă și multe alte limbi mai exotice ale lumii este un lucru ușor.

În afară de diferențele menționate în această listă, XML este foarte asemănător cu HTML din punctul de vedere al marcării etichetelor, al argumentării atributelor și al trecerii conținutului între perechi de etichete. Dacă scrieți HTML *curat*, atunci conversia HTML-ului dumneavoastră la XHTML-ul bazat pe XML este atât de simplă încât este posibil să lăsați un calculator să facă acest lucru în locul dumneavoastră. Desigur, XML nu se limitează la limbaje care seamănă cu HTML, deci structura documentului dumneavoastră este limitată numai de structura arborescentă necesară și de propria dumneavoastră imaginație.

Sugestie de la LeeAnne.com Words to weave by

Pentru sistemele Windows, sunt greu de depășit puterea și funcționalitatea programului HTML-Kit, de la Chami.com (`http://www.chami.com/`), care utilizează excelentul program Tidy al lui Dave Raggett pentru a curăța și, opțional, pentru a converti codul dumneavoastră la XML curat. El inserează toate acele etichete de încheiere plictisitoare și sintaxa etichetei speciale de încheiere `EMPTY`, aranjează etichetele care nu se imbrică în mod corespunzător și multe altele. Tidy este disponibil pentru Mac cu o portare a lui Terry Teague la `http://www.geocities.com/SiliconValley/1057/tidy.html`, iar pe Web există multe alte variante pentru UNIX. Mai există și o portare Java pentru entuziaștii O-O (orientare spre obiecte). Pentru detalii actualizate despre aceasta și despre alte versiuni, vizitați pagina Web a lui Dave Raggett de la W3C, la `http://www.w3.org/People/Raggett/tidy/`.

Folosind deprinderile pe care probabil că le aveți azi, puteți începe realizarea de documente XML simple chiar după câteva ore de exersare. Limbajul a fost proiectat astfel încât să fie transparent la utilizare pentru a putea fi înțeles și utilizat cu ușurință. Descrierile XML succinte sau complicate din majoritatea documentelor sunt greu de înțeles în efortul de a fi explicit într-un mod în care programatorii să îl poată translata cu ușurință în aplicații care să funcționeze. Mai este de învățat pentru obținerea unei adevărate perfecțiuni, dar nu este chiar așa de dificil.

Înainte de a începe dezmembrarea limbajului, ar trebui să priviți documentul XML ca întreg.

Definirea unui document XML ca întreg

Un document XML este o colecție de entități care pot fi sau nu analizate. Datele care nu sunt înțelese de un procesor XML, date binare sau date care au sens numai pentru alte aplicații, reprezintă *date neanalizate*. Datele înțelese de XML, fie ca și caractere fie ca marcaje, sunt *date analizate*.

Un document XML trebuie să fie construit corect. În Recomandarea XML 1.0 a W3C, citată aici cu exactitate, această situație este descrisă concis prin următoarele cerințe:

- Luat ca întreg, corespunde producției etichetate ca *document*.
- Satisface toate constrângerile de construire corectă din această specificație (Recomandarea XML 1.0).
- Fiecare entitate analizată, referită direct sau indirect în cadrul documentului, este construită corect.

Prima constrângere spune că pentru a fi construit corect, un document XML trebuie să respecte toate regulile care descriu un document în Recomandarea XML 1.0. Acele reguli spun în primul rând că un document XML trebuie să conțină un prolog și un singur element care formează elementul rădăcină al documentului, împreună cu comentarii opționale și instrucțiuni de prelucrare. De asemenea, ele mai spun că puteți atașa comentarii și instrucțiuni de prelucrare la sfârșitul documentului, dar, din păcate, analizorul XML nu are cum să spună dacă aceste comentarii și instrucțiuni de prelucrare atașate sunt sau nu asociate documentului. Deoarece ele pot să fie plasate după eticheta de încheiere, un analizor XML nu poate să spună nici măcar dacă toate instrucțiunile de prelucrare și comentariile atașate au fost sau nu recepționate. Aceasta contravine regulii generale din XML care spune că un analizor trebuie să fie capabil să indice dacă un document este sau nu complet. Dacă utilizați instrucțiuni de prelucrare sau comentarii, treceți-le în prolog unde sunt mult mai în siguranță și nu se pot pierde.

Cea de a doua constrângere spune că documentul respectă constrângerile construirii corecte descrise în document. Aceste constrângeri sunt cercetate în secțiunea următoare, „Constrângerile construirii corecte". Una dintre constrângerile construirii corecte este că entitățile analizate recursiv sunt interzise. Recursivitatea din această interdicție se referă la formarea unei bucle de entități în care o entitate se încorporează pe sine însăși sau o altă entitate care se încorporează pe sine însăși indiferent de nivelul de indirectare. Aceasta mai înseamnă și că un document nu se poate referi la el însuși, nici măcar indirect printr-o entitate externă. El se poate referi la o entitate externă numai dacă aceasta nu se referă la ea însăși, chiar și indirect. Este posibil ca analizoarele care nu validează să nu depisteze această eroare, dar aceasta este totuși o eroare. Din punct de vedere logic este evident că dacă documentul A conține documentul B, definindu-l pe B ca și când l-ar conține pe A se generează o buclă infinită. Bucla infinită este interzisă.

Notă În termeni XML, construcția corectă este o altă modalitate de a spune că un document XML formează un arbore, sau o ramură a unui arbore, adică este complet în sine. Acest lucru este necesar deoarece XML vă permite să construiți documente mai mari din unele mai mici și reprezintă o cheie a posibilității de utilizare a XML pe Web. Pe parcursul acestui capitol veți descoperi și alte reguli formale.

Deși construirea corectă poate fi considerată suficientă deoarece un document construit corect are un DTD care îl descrie, pot fi construite un număr infinit de DTD-uri care, de asemenea, să îl descrie. Deci, pentru validitate totală, un DTD asociat este necesar. Despre DTD-uri veți învăța mai multe în Capitolul 3, „Definițiile tipurilor documentului XML".

Producția documentului este definită în numai două afirmații, preluate din nou din Recomandarea XML 1.0:

- Conține unul sau mai multe elemente.
- Există un singur element, numit *rădăcină* sau *element document*, care să nu aibă nici o parte a sa în conținutul oricărui alt element. Pentru toate celelalte elemente, dacă eticheta de pornire se află în conținutul unui alt element, atunci și eticheta de încheiere se află în conținutul aceluiași element. Exprimat mai simplu, elementele delimitate de etichete de pornire și încheiere sunt imbricate corect unele în altele.

Prima afirmație spune că într-un document trebuie să existe cel puțin un element sau, altfel spus, un document construit corect nu poate fi vid.

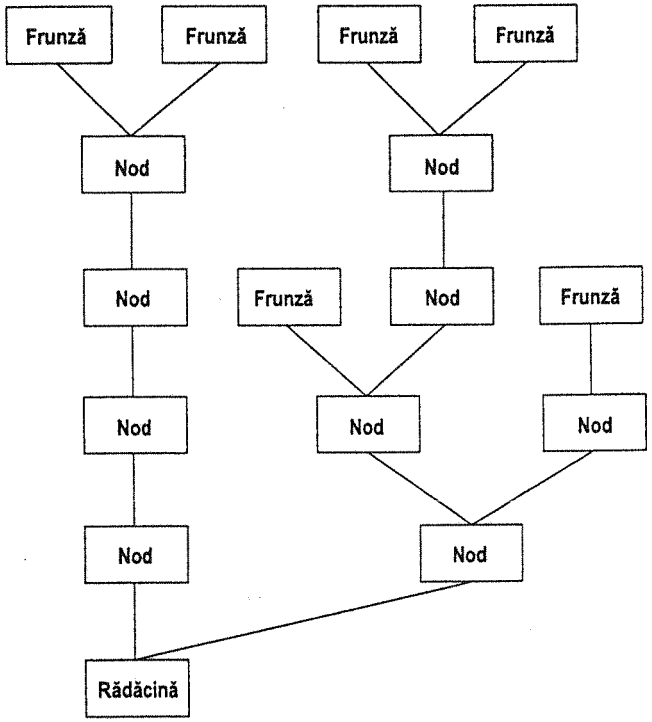
Cea de a doua afirmație spune că, într-un sens restrâns, un document trebuie să fie un arbore. El nu poate fi o rețea conectată arbitrar sau să aibă orice altă topologie care nu se reduce la un arbore simplu. El trebuie să fie complet pentru a putea vedea diferența dintre o descărcare reușită și una parțială.

Notă Din punct de vedere tehnic, un *arbore* este un graf conex care nu conține circuite. Cu alte cuvinte, un arbore se ramifică de la rădăcina sa fără a se mai conecta la el însuși, deci nu conține muchii multiple sau bucle. Orice lucru care conține bucle sau muchii multiple nu este un arbore ci altceva, și nu îl puteți reprezenta în XML. Un efect secundar interesant al acestui lucru este acela că într-un arbore puteți alege arbitrar un punct, să îl „aranjați" puțin și să transformați un nod într-o rădăcină, reorganizând arborele într-un alt arbore, cu o altă ordine de parcurgere. Aceasta ilustrează caracterul capricios al schemelor de clasificare.

O descărcare parțială este posibilă în HTML, deoarece HTML nu are nevoie de eticheta de încheiere `</html>`, sau nu are nevoie de aproape nici o etichetă de încheiere. Uneori, browserul poate detecta întreruperea, dar acest lucru nu este garantat. Aceasta înseamnă că un document parțial se poate deghiza într-unul complet, iar utilizatorul nu are de unde să știe acest lucru până când în text nu apare o eroare evidentă. XML previne aceste probleme, lucru care poate fi important în cazul în care un utilizator reclamă ulterior că o convenție de acordare a licenței, de exemplu, nu a fost afișată în întregime. Insistând asupra unui arbore complet, un astfel de exemplu fiind prezentat în Figura 2.1, se elimină aceste probleme posibile.

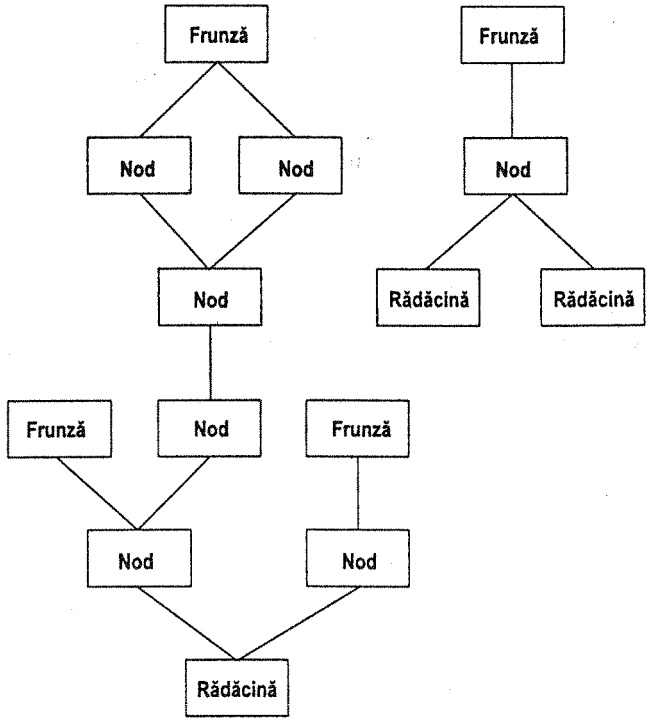
Pe de altă parte, un graf care nu formează un arbore nu poate fi transformat într-un document XML decât dacă graful poate fi simplificat, eliminând toate caracteristicile care nu sunt reprezentative pentru arbori. În Figura 2.2, de exemplu, graful din stânga poate fi simplificat prin eliminarea unei căi de la frunza de sus către un nod. În aceeași figură, grafului din partea dreaptă trebuie să i se elimine o rădăcină deoarece un document XML poate avea o singură rădăcină.

FIGURA 2.1
Această figură reprezintă un arbore construit corect. Din structura reprezentată de acest arbore puteți realiza un document XML.



Trebuie să fiți atent și la faptul că deseori arborii sunt reprezentați de sus în jos, cu rădăcina în partea de sus și ramurile crescând în jos. Aceasta corespunde obișnuinței noastre de a citi paginile de sus în jos, deci primul lucru pe care îl întâlnim la acest arbore inversat este rădăcina, așa cum ar face și un analizor XML, iar parcurgerea în jos a paginii ne va duce tot mai adânc în „frunzișul" arborelui.

FIGURA 2.2
Această ilustrație reprezintă două grafuri care nu sunt arbori. Ele nu pot fi transformate în documente XML.



Constrângerile construirii corecte

Pe lângă proprietățile de bază necesare într-un document XML, prezentate în secțiunea anterioară, „Definirea unui document XML ca întreg", un document XML trebuie să îndeplinească anumite criterii suplimentare, numite constrângeri. Lista următoare descrie constrângerile de construire corectă:

- Entitățile parametru din submulțimea internă pot să apară numai acolo unde pot să apară marcaje. Ele nu pot să apară în interiorul marcajelor. Aceasta este o regulă complet arbitrară și a fost introdusă pentru a ușura operația de analizare a DTD-urilor interne (vezi Capitolul 3).
- Numele din eticheta de încheiere a unui element trebuie să coincidă cu numele din eticheta de pornire. Acesta este un lucru evident. Puțini dintre noi se vor aștepta să poată încheia o etichetă `<cite>` cu o etichetă `</citation>`.
- Un nume de atribut poate să apară o singură dată în cadrul aceleiași etichete de pornire sau într-o etichetă de element vid. Din nou, este destul de evident. Ce credeți că veți face cu o linie de cod diformă ca aceasta?

```
<image src="unu.gif" src="doi.gif" />
```

- Valorile atributelor nu pot conține referiri directe sau indirecte la entități externe. Acest lucru este mai subtil și a fost introdus pentru a ușura viața procesoarelor XML. Într-un mediu în care este posibilă codificarea arbitrară a caracterelor în entități externe, manevrarea lor corectă într-un atribut ar fi dificilă.

- Textul de înlocuire al unei entități referite direct sau indirect în valoarea unui atribut (alta decât `"<"`) nu poate să conțină un caracter <. Aceasta este pentru simplitate și pentru tratarea erorilor. Dacă permiteți un < care nu este într-o secvență escape în interiorul unui atribut, atunci va fi dificil să găsiți niște ghilimele finale care lipsesc. De asemenea, deoarece oricum trebuie să includeți < în secvențe escape în text, tratarea sa diferită în interiorul valorii unui atribut este o inconsecvență.
- Caracterele referite folosind referiri caracter trebuie să fie caractere permise. Cu alte cuvinte, nu puteți ascunde caractere, care altfel nu ar fi permise, prin indirectare sau definindu-le prin echivalentul lor numeric. De exemplu, `&#x0000;` nu este un caracter permis, indiferent de modul în care îl referiți.
- Într-un document care nu are un DTD, un document care are doar o submulțime DTD internă care nu conține referințe la entități parametru sau un document cu o valoare `"standalone='yes'"` în declarația XML, numele dat într-o referință a unei entități trebuie să se potrivească cu cel din declarația entității. O excepție este aceea că documentele construite corect nu trebuie să declare nici una dintre următoarele entități: `&`, `<`, `>`, `'` sau `"`.
  În principiu, declararea unei entități parametru trebuie să preceadă orice referire la ea, dar există câteva situații în care un procesor XML care nu validează se va opri din prelucrarea declarațiilor entităților. Deci, dacă procesorul care nu validează este convins că toate declarațiile au fost citite și prelucrate, atunci el poate afișa o eroare referitoare la construirea corectă și să întrerupă prelucrarea în cazul în care găsește o entitate nedeclarată. Pe de altă parte, dacă a fost urmată orice altă cale prin care procesorul XML care nu validează se oprește din prelucrarea entităților, atunci întâlnirea unei entități nedeclarate nu este o eroare. Acesta este un mod complicat de a spune că procesoarele XML care nu validează pot să depisteze sau nu entități nedeclarate în funcție de situație.
- Referința la o entitate nu poate să conțină numele unei entități neanalizate. Pe scurt, nu puteți trânti date binare în mijlocul unui text fără a avea declarat un fel de mecanism de manipulare. Deci, următorul cod este permis, iar valoarea este transferată agentului utilizator sau browserului dacă și numai dacă reprezintă o entitate externă neanalizată care a fost deja declarată ca o notație:

```
<image &pozamea; />
```

- O entitate analizată nu are voie să conțină o referință recursivă la ea însăși, nici direct și nici indirect. Cu toate că este posibil ca autorilor de dicționare să le placă să declare că un steag este un drapel iar un drapel este un steag, XML nu vă permite să faceți așa ceva.
- Referințele la entități parametru pot să apară numai în DTD. Cu alte cuvinte, nu puteți include date de prelucrare în documentul final și să vă așteptați ca acestea să însemne ceva. La fel, nu vă puteți aștepta ca inserând o instrucțiune C, de exemplu `printf("Hello, world"\n);`, în pagina dumneavoastră dactilografiată, aceasta să fie înlocuită cu o valoare și să se efectueze și un retur de car. Pe de altă parte, cu toate că este o eroare de logică dacă dumneavoastră vă *așteptați* ca asta să se întâmple, un astfel de text nu va strica nimic, la fel cum tipărirea liniei de cod C de mai sus în textul dumneavoastră nu va genera o eroare. Deoarece % este doar un caracter și nu trebuie să fie inclus în document prin secvențe escape, e greu de imaginat cum poate fi interceptată o astfel de „eroare". Deși este un fapt interesant, aceasta este o instrucțiune nulă din punct de vedere al erorilor de prelucrare.

Prologul: Declarația XML

Fiecare document XML ar trebui să înceapă cu o declarație XML care identifică fișierul ca fiind un fișier XML și identifică, de asemenea, versiunea XML utilizată în documentul respectiv. Caracterul opțional este datorat numai faptului că pe Web există multe fișiere HTML și SGML care corespund perfect unui XML construit corect. De ce să stricăm ceea ce funcționează deja? De asemenea, declarația XML este locul unde declarați codificarea dumneavoastră și dacă documentul este sau nu autonom. Ordinea din fragmentul următor este obligatorie, cu toate că atributele `encoding` și `standalone` sunt ambele opționale:

```
<?xml version="1.0" encoding=" ISO-8859-1" standalone="yes">
```

Codificările vă permit să identificați pe care dintre multiplele seturi de caractere intenționați să le utilizați în document. Acest lucru este important deoarece, spre deosebire de HTML, care implică ASCII și vă obligă să folosiți nume ASCII, XML permite vorbitorilor de Hindi, de exemplu, să utilizeze codificarea Devanagari și să facă textul lor și mediul de editare lizibil și pentru persoanele obișnuite care pot fi autori XML. Sau, un autor chinez poate să prefere caracterele chinezești în etichete și în conținut. Cu câteva limitări bazate pe regulile limbajului în sine, puteți utiliza moduri de scriere și ideograme atât în numele etichetelor cât și în conținut.

Ca un exemplu simplu referitor la tipurile de limitări bazate pe regulile limbajului, gândiți-vă la literele indice superior din limba engleză „st", din cvasi-abrevierea „1st". Literele îl modifică de fapt pe „1", astfel încât dumneavoastră să știți că trebuie să pronunțați „first" (primul). Nu există în limba engleză (sau într-o altă limbă europeană) situație în care să aveți voie să folosiți această construcție la începutul unui „cuvânt". Deci, „stwhile" nu se extinde legal în cuvântul „erstwhile".

În mod asemănător, „St Ives" trebuie pronunțat „Saint Ives". În XML, conform regulilor, puteți defini un element pe nume `<St>`, dar deoarece literele indice superior nu sunt permise la începutul cuvintelor în nici o limbă europeană, nu puteți defini un element care să se numească `<tmisbehaving>` (și să se pronunțe „ain't misbehaving"), deoarece ar încălca regulile limbajului nostru. Într-un mod interesant, cu toate că nu puteți utiliza cifre obișnuite la începutul unui nume, puteți folosi cifre romane, chiar dacă cifra romană „3" este definită ca fiind un singur caracter, „III". Aceasta face să funcționeze construcțiile relativ neobișnuite, cum ar fi „XIVth Century".

Atributul `standalone` vă permite să dezactivați, dacă doriți, un DTD extern. Acest lucru poate avea și alte efecte, dar este dificil de spus în câteva cuvinte care vor fi acestea. Veți învăța mai multe despre acest atribut în secțiunea „Documente autonome" din acest capitol.

Codificări de caractere

Este aproape sigur că cea mai uzuală codificare de caractere pentru engleza americană (și pentru oricare altă limbă) este UTF-8, care este și cea prestabilită. Acesta este setul de caractere Unicode extins, utilizat adesea pe sistemele UNIX. Este o codificare de lungime variabilă, utilizând între 1 și 5 octeți de date pentru fiecare caracter, fapt ce poate ridica probleme în Java și în alte medii. Dacă sistemul dumneavoastră cere o lungime de octeți fixă, UTF-16 care utilizează, ca și Java, doi octeți pentru un caracter, este o alegere bună, deoarece procesoarele XML trebuie să fie capabile să detecteze

fiecare codificare, fără a fi nevoie de nici o declarație. UTF-8 și UTF-16 sunt singurele codificări pe care analizoarele XML sunt *obligate* să le recunoască și sunt suficiente pentru utilizarea generală a oricărei limbi de pe pământ.

Pe de altă parte, majoritatea sistemelor Windows nu utilizează UTF-8, ci ISO-8859-1, care este setul de caractere ISO-Latin-1 utilizat pentru engleza britanică și cea americană și pentru majoritatea limbilor din Europa de Vest. Aceasta nu este o valoare prestabilită, deci ar trebui specificată în declarația dumneavoastră de codificare, astfel:

```
<?xml version="1.0" encoding="ISO-8859-1">
```

Altfel, este posibil ca anumite caractere să nu fie mapate corect la UTF-8 și să producă probleme de lizibilitate în documentul dumneavoastră. În general, dacă aveți de gând să realizați texte în primul rând în engleză, în care să aveți întâmplător un cuvânt sau o frază într-o altă limbă europeană, cam aceasta ar fi tot ce trebuie să știți despre codificarea caracterelor.

Informațiile următoare prezintă interes deosebit pentru cei a căror limbă de bază este alta decât engleza și care probabil că vor recunoaște codificările produse de aplicațiile lor de prelucrare de texte sau le vor deduce din context. Dacă nu aveți de gând să realizați documente utilizând aceste codificări, puteți sări peste întreaga secțiune „Alte codificări".

Alte codificări

Alte codificări obișnuite sunt ISO-10646-UCS-2 și ISO-10646-UCS-4, care pot fi folosite pentru două dintre codificările Unicode/ISO/IEC 10646 utilizate cel mai des pentru unele limbi europene.

Valorile ISO-8859-1, ISO-8859-2 și celelalte până la ISO-8859-10 pot fi utilizate pentru diversele părți ale ISO 8859 (fonturi pe 8 biți) utilizate pentru Latin-1 (Europa de Vest), Latin-2 (Europa de Est), Latin-3 (Europa de Sud), Latin-4 (Europa de Nord), Cyrillic (chirilică), Arabic (arabă), Greek (greacă), Hebrew (ebraică), Latin-5 (turcă) și Latin-6 (nordică), în această ordine. Valorile ISO-2022-JP, Shift_JIS și EUC-JP ar trebui utilizate pentru codificările limbii japoneze ale JIS X-0208-1997. Fiecare procesor XML ar trebui să recunoască măcar aceste codificări, dar este posibil să nu se întâmple așa. Procesoarele XML pot recunoaște și alte nume de codificări, cum ar fi ASCII, US-ASCII sau EBCDIC-US. Recomandarea XML 1.0 a W3C sugerează că acele codificări de caractere înregistrate la Internet Assigned Numbers Authority (IANA – Autoritatea de atribuire a numerelor în Internet), altele decât cele prezentate mai sus, ar trebui referite utilizând numele lor înregistrate.

Sugestie de la LeeAnne.com Words to weave by

Aceste nume înregistrate nu sunt sensibile la tipul literelor, deci procesoarele care încearcă să le recunoască ar trebui să o facă într-un mod care să nu depindă de tipul literelor.

Dacă alegeți o anumită codificare a caracterelor, alta decât cele prestabilite, UTF-8 sau UTF-16, care adresează potențial întregul spațiu de caractere Unicode/ISO de mai mult de un milion de caractere, ar trebui de asemenea să vă asigurați că seturile de entități importante sunt și ele importate în documentul dumneavoastră, pentru claritate și pentru ușurința utilizării (dacă utilizați un procesor de texte sau un mediu de creare în limba

engleză). Acest lucru îl veți face de obicei pentru ISO-8859-1 și pentru caracterele speciale definite pentru HTML și XHTML. Pentru majoritatea sistemelor MS-Windows din Europa de Vest și din cele două Americi, ar trebui să alegeți o codificare de caractere ISO-8859-1, setul de caractere Latin-1 utilizat de obicei de Windows. Veți utiliza declarațiile `ENTITY` care arată astfel:

```
<!-- Latin-1 characters for XHTML -->
<!ENTITY % HTMLlat1 PUBLIC "-//W3C//ENTITIES Latin1//EN//HTML"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent">
   %HTMLlat1;
<!-- Special characters for XHTML -->
<!ENTITY % HTMLspecial PUBLIC "-//W3C//ENTITIES Special//EN//HTML"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml-special.ent">
    %HTMLspecial;
<!-- Mathematical, Greek and Symbolic characters for XHTML -->
<!ENTITY % HTMLsymbol PUBLIC
       "-//W3C//ENTITIES Symbols//EN//HTML"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml-symbol.ent">
    %HTMLsymbol;
```

Despre declararea caracterelor veți învăța mai multe pe parcursul acestui capitol, dar din această discuție ar trebui să rămâneți cu ideea că pentru documentele obișnuite, în limba engleză, utilizând software de creare bazat pe Windows, veți dori probabil să declarați explicit setul de caractere ISO-8859-1 pentru a vă asigura că setul de caractere ISO-Latin-1 va interpretat corect. Cu toate că literele de pe tastaturile englezești obișnuite sunt echivalente în UTF-8 și ISO-8859-1, anumite caractere accentuate și speciale sunt diferite, iar setul de caractere ISO-8859-1 este mult mai mic.

Dacă utilizați un procesor de texte sau un software de creare pentru alte limbi decât engleza și mai ales dacă utilizați un sistem de operare care nu este european, atunci ar trebui să vă lămuriți care codificări sunt generate, pentru ca în declarația dumneavoastră să puteți apela la codificarea corespunzătoare.

Probabil că aici sunt mai multe informații decât ar dori să afle majoritatea oamenilor despre codificările de caractere, dar atunci când va trebui să realizați un document lizibil, de exemplu în rusă, veți avea foarte mare nevoie de ele. Situl Web Roman Czyborra conține legături la mult mai multe informații, inclusiv imagini ale celor mai uzuale seturi de caractere: `http://czyborra.com/charsets/iso8859.html`.

Documente autonome

În conformitate cu Recomandarea XML 1.0 a W3C, „Documentele autonome nu au declarații de marcare externe care să afecteze informația XML transferată de procesorul XML unei aplicații".

Acesta este un mod extraordinar de succint și obscur de a spune că `standalone="yes"` înseamnă că:

- Într-un DTD extern nu există valori prestabilite declarate ale atributelor care să nu fie setate explicit în document.
- Nu sunt utilizate alte entități în afară de `&`, `<`, `>`, `'` și `"` care nu au fost declarate local sau poate citite prin referință dintr-un fișier.

- Nu există elemente care să aibă ca și conținut numai spații albe de orice natură.
- Nu există atribute externe care trebuie normalizate, deci conținutul atributelor nu poate conține spații albe, caractere sau referințe la entități.

Aceasta *nu* înseamnă că documentul nu are nimic extern. Poate are. Aceasta înseamnă în special că, indiferent de locul în care un procesor XML care nu validează se oprește din citirea documentelor externe, se oprește prelucrarea *tuturor* declarațiilor.

Toate aceste lucruri le puteți face dacă și numai dacă le puneți în submulțimea DTD-ului intern.

Datele externe care nu sunt marcaje nu fac subiectul acestei afirmații. Deci, puteți avea fișiere grafice, fișiere text incluse și orice altceva, atât timp cât ele nu sunt marcaje și le-ați declarat în submulțimea DTD-ului intern.

După toate acestea, procesorul XML nu trebuie să anunțe aplicația dacă documentul este sau nu autonom. De fapt, procesorul nu trebuie să facă nimic cu această informație și nici să se comporte într-un anumit fel atunci când întâlnește această informație.

În mod fundamental, proiectantul DTD-ului trebuie să descifreze dacă documentul creat utilizând acel DTD poate fi sau nu autonom și să comunice acest lucru oamenilor, inclusiv autorilor. Autorii care știu că DTD-ul a fost proiectat pentru a fi autonom sau cei care au convertit un document care nu a fost proiectat să fie autonom în acest format, pot insera `standalone="yes"` în declarația lor XML ca o documentație a acelui fapt:

```
<?xml version="1.0" standalone="yes" ?>
```

Documentele care nu sunt autonome pot fi convertite prin algoritmi la documente autonome, automat, presupunând că este disponibilă o facilitate care să realizeze acest lucru, sau manual în caz contrar.

Construirea prologului unui document XML: Declarația tipului documentului (Document Type Declaration)

Prologul unui document XML conține mai multe instrucțiuni. Prima, declarația XML, afirmă că documentul următor este XML. Cea de a doua, declarația tipului documentului (Document Type Declaration), este metoda pe care o utilizați pentru a identifica definiția tipului documentului (Document Type Definition – DTD) folosită de un anumit document. Faptul că acronimul DTD poate fi aplicat la Document Type Declaration este o coincidență nefericită. DTD se referă numai la Document Type Definition. Într-un document XML poate exista o singură declarație a tipului documentului, deci este introdusă chiar în instanța documentului. Deoarece pot fi combinate mai multe DTD-uri pentru a forma un singur document, aceasta permite controlul încărcării DTD-urilor în fiecare document.

Declarația tipului documentului (`DOCTYPE`) are două părți, ambele opționale. Prima se referă la un DTD extern și utilizează cuvintele cheie `PUBLIC` sau `SYSTEM` pentru a identifica o intrare dintr-un catalog, respectiv un URI. În cazul în care cataloagele nu sunt implementate în procesorul dumneavoastră XML, puteți specifica ambele părți deodată, fără cel de al doilea cuvânt cheie:

```
<!DOCTYPE nume-document PUBLIC "{catalog id}">
```



```
<!DOCTYPE nume-document PUBLIC "{catalog id}" "{uri}">
<!DOCTYPE nume-document SYSTEM "{uri}">
```

Cea de a doua parte opțională a declarației `DOCTYPE` vă permite să introduceți submulțimea internă a unui DTD direct în documentul dumneavoastră. Există restricții severe cu privire la genul de informație pe care o puteți introduce în DTD-ul intern, dar oricum puteți face destul de multe. Submulțimea internă a unui DTD este încadrată între paranteze drepte, astfel:

```
<!DOCTYPE nume-document [ {declaratiile DTD-ului intern} ]>
```

De asemenea, cele două pot fi combinate, permițându-vă să adăugați anumite tipuri de declarații și entități aproape în orice mod:

```
<!DOCTYPE nume-document PUBLIC "{catalog id}" "{uri}" [ {declaratiile DTD-ului intern} ]>
```

Pentru claritate, submulțimea internă este evidențiată de obicei cu retururi de car ca mai jos:

```
<!DOCTYPE nume-document PUBLIC "{catalog id}" "{uri}" [
 {declaratiile DTD-ului intern}
]>
```

Declarația `DOCTYPE` trebuie să utilizeze numele elementului rădăcină al DTD-ului, fie acesta intern sau extern, ca fiind câmpul etichetat `nume-document` din exemplele anterioare. Deci dacă numele elementului rădăcină al DTD-ului dumneavoastră este `Dave`, declarația dumneavoastră `DOCTYPE` ar trebui să înceapă astfel:

```
<!DOCTYPE Dave … >
```

Manualul sau foaia dumneavoastră de codare vă indică ce să treceți în `DOCTYPE` dacă sunteți un autor. Dacă sunteți un proiectant DTD, atunci ar trebui să furnizați un astfel de manual sau o foaie de codare fiecărui autor.

De asemenea, puteți crea un DTD master care apelează părțile DTD de care aveți nevoie, lucru care seamănă cu comenzile dintr-un meniu. Atunci când aveți o combinație de funcționalități care vă permite să creați structura documentului de care aveți nevoie, puteți publica DTD-ul rezultat și să scăpați de neplăcerea de a-l face iar și iar pentru fiecare document nou. Puteți afla mai multe despre DTD-uri în Capitolul 3.

Crearea corpului documentului

Un document XML este alcătuit din text, care de obicei este format dintr-un amestec de marcaje și date caracter. Prologul conține numai marcaje, dar nu aceasta este partea interesantă deoarece dumneavoastră aveți nevoie de date care să însoțească marcajele care, altfel, nu ar fi decât niște cutii goale ce așteaptă să fie umplute. Corpul documentului dumneavoastră conține aproape tot ceea ce contează din perspectiva unei aplicații (sau a unui om), împrăștiat generos în cadrul marcajelor.

Date caracter

Un DTD poate declara multe tipuri de date care să poată fi utilizate în documentul dumneavoastră, dar tipul de date prestabilit este întotdeauna `CDATA`, pentru date obișnuite de tip caracter. Foaia sau manualul de codare vă indică ce tip de dată poate fi introdus în fiecare atribut sau câmp cu conținut element.

Presupunând că tipul de date este CDATA, în câmp puteți pune aproape orice doriți atât timp cât nu conține un marcaj în afara unei secvențe escape.
Este pe deplin posibil să construiți un DTD care să nu conțină text în interiorul elementelor. În schimb, puteți pune datele semnificative în interiorul atributelor asociate fiecărui element, care pot fi toate declarate ca fiind vide sau având numai conținut element. Acest lucru se face uneori pentru a converti un document utilizând un standard de codificare cum ar fi MARC, care este în esență un format binar, la XML.

Marcajul

Marcajul este format din ansamblul de date non-caracter dintr-un fișier XML. Diversele forme pe care le poate lua un marcaj sunt prezentate în Tabelul 2.1.

Tabelul 2.1 Sintaxa marcajului XML

Tipul marcajului	Sintaxa marcajului
etichete de pornire	`<numeElement [atribute] > …`
etichete de încheiere	`… </numeElement>`
etichete element vid	`<numeElement [atribute] />`
referințe entitate	`&numeEntitate; sau %parametruNumeEntitate;`
referințe caracter	`&#numarZecimal; sau &#xnumarHexa;`
comentarii	`<!-- comentariu -->`
delimitatoarele secțiunii CDATA	`<![CDATA[ informatii cdata ]]>`
declarațiile tipului documentului	`<!DOCTYPE nume IDextern? [informatiiDTD]>`
instrucțiuni de prelucrare	`<?IDprocesor date ?>`
declarația XML	`<?xml version encoding standalone ?>`

Toate celelalte sunt date de tip caracter.
Marcajul începe întotdeauna fie cu caracterul <, caz în care se încheie întotdeauna cu caracterul >, fie cu caracterul &, caz în care se încheie întotdeauna cu caracterul ;. În secțiunile următoare ale acestui capitol vom studia diferitele tipuri de marcaje.

Formarea structurilor logice în XML

Imbricarea elementelor este singurul mecanism utilizat pentru a arăta structura logică dintr-un document XML. Etichetele de pornire și încheiere din fluxul de text spun procesorului XML că a fost întâlnit un nod.
Dacă procesorul XML întâlnește o altă etichetă de pornire înainte de eticheta de încheiere corespunzătoare, atunci procesorul știe că aceasta este fie un nod nou în arbore, fie o frunză. Dacă nu găsește o nouă etichetă de pornire și întâlnește o etichetă de încheiere, atunci procesorul știe că aceasta este o frunză și că poate acționa iterativ la acel nivel al arborelui până când întâlnește o altă etichetă de pornire sau de încheiere. Prelucrarea acționează treptat, bazându-se pe această regulă simplă. Dacă procesorul validează documentul, atunci fiecărui nod i se poate asocia o regulă care să determine ce tip de conținut poate apărea în el. O etichetă vidă este, prin definiție, o frunză, deoarece nu poate avea nici un conținut.

Notă Majoritatea structurilor de date conținute într-un document XML pot fi accesate secvențial fără a construi structura în memorie. O etichetă de pornire începe un nod sau o frunză, iar eticheta de încheiere corespunzătoare îl (o) încheie. Orice etichete întâlnite între o etichetă de pornire și eticheta de încheiere corespunzătoare ei încep un nou nod sau o frunză nouă. Acest principiu reprezintă baza lui SAX și a altor procesoare XML conduse prin evenimente (vezi Capitolul 11, „SAX (The Simple API for XML) – API simplu pentru XML").

Restul structurii logice a documentului este definit prin atributele asociate fiecărui element. În plus, structura logică poate diferi în funcție de conținutul secțiunilor condiționale din cadrul documentului sau al componentelor sale.

Cum formează XML structurile fizice

Imbricarea entităților este singurul mecanism utilizat pentru a arăta structura fizică dintr-un document XML. Definițiile de entități întâlnite în fluxul de text îi comunică procesorului XML că a fost găsită o altă entitate.
Există multe tipuri de entități, de la entitățile mici care formează caractere individuale, cum ar fi: (spațiu) sau &sp; (spațiu), până la entitățile externe care vă permit să încorporați în documentul dumneavoastră porțiuni din alte documente XML sau să includeți într-un document referințe la date neanalizate, cum ar fi fișiere multimedia pentru redarea ulterioară de către un agent utilizator.
Un document XML este o colecție de astfel de entități. Fiecare dintre aceste subentități trebuie să fie completă în sine. Aceasta înseamnă că, deoarece structura documentului ca întreg trebuie să fie un arbore simplu, fiecare subentitate trebuie să fie un singur nod sau trebuie să fie, de asemenea, un arbore simplu. Structuri mai mari puteți construi prin adăugarea nodurilor sau arborilor subentitate ca porțiuni ale arborelui dumneavoastră mai mare.
Dacă studiați Figura 2.1, prezentată anterior, puteți împărți acea diagramă în subarbori numai dacă, luând un creion și încercuind toate elementele arborelui pe care vi l-ați propus, tăiați o singură linie, ramura care unește grupul dumneavoastră de elemente cu arborele principal. Dacă tăiați mai multe linii, nu puteți forma un subarbore corect. Aceasta înseamnă că trebuie să includeți un singur nod, cel mai de jos, care va reprezenta rădăcina noului dumneavoastră subarbore.
Dacă documentele dumneavoastră conțin mai mulți subarbori în fișiere diferite, fiecare fișier subarbore trebuie să fie un arbore document XML complet în sine.
Priviți cele două grafuri conexe prezentate anterior în Figura 2.2. Vedeți că există fie două rădăcini, ceea ce, vă amintiți, este interzis în XML, fie un circuit simplu (o buclă) în graf care îl împiedică să fie o structură XML. Sunt, de fapt, substructuri ale acelui arbore care par să fie arbori simpli; totuși, acea buclă este un defect fatal. Ca întreg, aceste structuri nu pot fi transformate în documente XML. În primul caz, prin rearanjarea structurii, puteți probabil să o transformați într-un arbore simplu, dar o rădăcină va deveni o frunză în noua structură.
În cel de al doilea caz, puteți izola o mare parte a structurii ca fiind un arbore simplu și să o transformați într-o structură XML. Dar, va trebui să găsiți o altă metodă de a reprezenta acea parte care nu este un arbore, în afară de cazul în care eliminați una din căile circulare și o transformați într-un arbore simplu.

Sugestie de la LeeAnne.com Words to weave by

Un graf conex, cu un circuit, poate fi aranjat prin eliminarea unei părți a circuitului, după care va rezulta un arbore complet.

Ca o soluție posibilă, structura care ciclează poate fi declarată ca notație și poate fi transferată pentru manipulare unei alte aplicații.

Alternativ, ca o idee primitivă și aproape trivială, o hiperlegătură ar putea rezolva problema dumneavoastră, în funcție de ceea ce încercați să comunicați prin acea structură. La o hiperlegătură, săriți din document într-un alt loc, metaforic vorbind, deasupra planului hârtiei pe care este tipărit documentul dumneavoastră. Puteți ateriza oriunde, inclusiv chiar în document, realizând în acest mod, de altfel singurul posibil, isprava de a sări din frunză în frunză sau într-un alt arbore.

O hiperlegătură reprezintă astfel o meta-informație despre structura arborelui sau despre structura altor arbori și care stă într-o oarecare măsură în exteriorul arborelui în sine.

Etichete de pornire și etichete de încheiere

În XML sunt utilizate două tipuri de etichete, etichete cu conținut și etichete vide. Etichetele cu conținut trebuie să aibă o etichetă de pornire și o etichetă de încheiere. Eticheta de pornire conține numele elementului încadrat între paranteze unghiulare, având opțional atribute ca argumente. Eticheta de încheiere conține numele elementului precedat de caracterul slash, totul fiind încadrat între paranteze unghiulare. În eticheta de încheiere nu puteți trece atribute. Codul următor reprezintă o etichetă cu conținut:

```
<titlu subtitlu="O calatorie acasa">Dus-intors</titlu>
```

Seamănă mult cu etichetele HTML standard și nu ar trebui să ridice alte probleme în afara celei de construire corectă, care cere ca ele să se imbrice într-adevăr una în cealaltă. Nu puteți avea etichete care să alterneze între ele ca în acest exemplu greșit construit:

```
<bold><italic>TEXT ACCENTUAT</bold></italic>
```

Cu toate că este o eroare obișnuită în HTML, XML este cu mult mai pretențios și nu va permite această construcție. Etichetele trebuie imbricate corect, după cum se vede în exemplul următor:

```
<bold><italic>TEXT ACCENTUAT</italic></bold>
```

Observați că acum etichetele se imbrică corect.

Atenție *Trebuie* să închideți fiecare etichetă care începe în contextul unei anumite etichete (sau al mai multor etichete) înainte de închiderea contextului etichetei respective.

Etichetele vide au disponibil un format special, cu toate că aceeași schemă, etichetă de pornire/etichetă de încheiere, poate fi utilizată atâta timp cât țineți minte că nu trebuie să puneți nici un fel de conținut între eticheta de pornire a elementului vid și eticheta de încheiere care urmează imediat după ea. De asemenea, poate că veți fi preocupat dacă este posibil ca documentul dumneavoastră XML să fie vizualizat cu un browser Web obișnuit, deoarece etichetele de încheiere pentru elementele care arată ca etichete HTML vide pot duce la blocarea browserului sau la un comportament ciudat al acestuia. Totuși, pentru utilizare generală, formatul special este mnemonic în sine, un avantaj deoarece dumneavoastră puteți *vedea* că eticheta este vidă și nu blochează aproape nici un browser.

De obicei, etichetele vide le porniți și le încheiați în cadrul aceleiași perechi de paranteze unghiulare, plasând după numele elementului și toate atributele sale posibile un spațiu, un caracter slash și apoi paranteza unghiulară închisă:

```
<image source="pozamea.jpeg" type="JPEG" />
```

Desigur, tipul trebuie declarat ca notație, dar toate acestea vor fi discutate în secțiunea „Entități neanalizate".

Normalizarea

Normalizarea este un cuvânt excentric pentru aducerea lucrurilor la cel mai mic numitor comun și punerea lor într-un fel de formă canonică. În contextul XML, el se referă la procesul rezolvării referințelor la entități în locații în care pot apărea astfel de referințe, la aranjarea rândurilor noi pentru a lua în considerare diversele metode de tratare a lor de către diferite sisteme de operare și la ordonarea altor lucruri care trebuie făcute în anumite situații.

Sugestie de la LeeAnne.com Words to weave by

Proiectantul rareori trebuie să se preocupe de normalizare, excepție făcând cazurile negative. Analizorul XML ar trebui să realizeze toate normalizările necesare, deci singurul lucru la care trebuie să se gândească creatorul documentului este dacă normalizarea va afecta sau nu datele sale atunci când se va face trecerea de la forma nenormalizată la cea normalizată și invers.

S-a dovedit că există două locuri în care pot fi întâlnite spații albe: în datele de tip caracter din cadrul documentului și în datele de tip caracter atribuite atributelor elementelor. În primul caz, în entitățile analizate este dificil să se facă distincție între spațiile albe „semnificative" și cele nesemnificative. Pentru proiectanți, cea mai bună cale pare să fie transferul către aplicație al tuturor spațiilor albe împreună cu cea mai bună presupunere a procesorului, bazată pe DTD, referitoare la datele care sunt în mod sigur nesemnificative și la cele nesigure. Acest transfer de răspundere are sens deoarece aplicația este cea mai în măsură să știe cum să procedeze cu spațiile albe suplimentare.

Atenție Procesorul XML poate numai să presupună care spațiu alb este semnificativ și care nu, bazându-se pe ceea ce a fost definit în DTD sau în orice alt limbaj de scheme utilizat. Aplicația dumneavoastră trebuie să fie pregătită să trateze presupunerile eronate.

Cu tratarea sfârșiturilor de linie, o altă formă de spații albe, există o altă problemă. Liniile noi sunt tratate diferit pe sisteme diferite. Alternativele generale sunt un salt la linie nouă (UNIX), un retur de car (MacOS) și atât returul de car cât și saltul la linie nouă (MS Windows). Un alt lucru obișnuit pentru aplicații este, de asemenea, inserarea de secvențe contradictorii cu oricare dintre acestea, în orice ordine, atunci când întâlnesc un fișier dintr-un sistem străin. W3C a hotărât că nu poate face totul și a ales un set rezonabil de reguli. Dacă analizorul întâlnește fie `
` (retur de car, salt la linie nouă), fie `
` (retur de car), el le înlocuiește cu `
` (salt la linie nouă), caracterul de linie nouă UNIX. Câțiva programatori Microsoft Windows au fost oarecum nemulțumiți.

Notă Microsoft şi Apple au ales mecanisme diferite pentru separarea liniilor, Microsoft alegând tehnica „curea şi bretele" obişnuită pentru teleimprimatoare, utilizând un retur de car şi apoi un salt la linie nouă pentru a indica o linie nouă. Apple a hotărât că saltul la linie nouă era redundant şi a considerat că un singur retur de car este suficient, modelat probabil după comportamentul unei maşini de scris obişnuite. UNIX a utilizat de la început un singur salt la linie nouă pentru a realiza acelaşi lucru, iar acesta a fost standardul convenit pentru XML. Deci, atât Microsoft cât şi Apple trebuie să folosească o logică specială pentru a manipula returile de car în documentele XML şi HTML.

În atribute, există o secvenţă de transformare standard şi apoi o prelucrare adăugată special pentru orice în afară de `CDATA`:

- Referinţele la caractere sunt prelucrate prin adăugarea caracterului referit la valoarea de ieşire a atributului.
- Referinţele la entităţi sunt procesate prin prelucrarea recursivă a textului de înlocuit al entităţii.
- Spaţiile albe, `#x20` (spaţiu), `#x0D` (retur de car), `#x0A` (salt la linie nouă) şi `#x09` (tab orizontal) sunt prelucrate prin adăugarea lui `#x20` (spaţiu) la valoarea normalizată a ieşirii, excepţie făcând faptul că se adaugă un singur `#x20` (spaţiu) pentru secvenţa `"#x0D#x0A"` (retur de car, salt la linie nouă), care este parte a unei entităţi externe analizate sau valoarea entităţii literale a unei entităţi analizate interne.
- Celelalte caractere sunt prelucrate prin adăugarea lor la valoarea normalizată a ieşirii.
- Dacă tipul de dată al atributului *nu* este `CDATA`, cel prestabilit, atunci se aplică încă o transformare. Spaţiile din faţă şi din spate sunt eliminate, iar spaţiile multiple sunt comasate într-un singur spaţiu.

Diferenţa dintre cele două tipuri de normalizare constă în faptul că puteţi să transferaţi în mod convenabil şiruri lungi într-un atribut, să restrângeţi linii astfel încât să încapă în pagină, iar în final conţinutul elementului să rămână relativ curat.

Tipuri de elemente

În mod surprinzător, dacă validaţi, nu este o eroare să utilizaţi un element de un tip care nu a fost declarat, cu toate că este posibil ca analizorul să emită un avertisment. De fapt, permiţând în interiorul unor elemente alte tipuri de elemente nedeclarate, indiferent de ceea ce spune modelul lor de conţinut, veţi fi capabil să suplimentaţi DTD-ul unui document cu elemente din alte zone de nume. Deci, tot ce aveţi de făcut este să utilizaţi elementul nedeclarat într-o manieră exactă, construită corect, identificând pe cât posibil zona de nume de unde provine. Deoarece aţi studiat deja ce înseamnă construirea corectă, aruncaţi o privire asupra unei situaţii mult mai interesante, un document valid.

Notă În termeni XML, a fi construit corect este un alt mod de a spune că acel element formează un arbore sau o ramură a unui arbore care este completă în sine. Acest lucru este necesar deoarece XML vă permite să construiţi documente mai mari din unele mai mici şi este un element-cheie pentru a fi capabil să utilizaţi XML pe Web. Alte reguli mai formale sunt discutate în alte secţiuni ale acestui capitol.

Fiecare element dintr-un document XML valid a fost definit în DTD-ul asociat acelui document prin declaraţia `DOCTYPE`. DTD-ul declară următoarele:

- Numele efective ale elementelor
- Regulile utilizate pentru a determina care elemente se pot imbrica în alte elemente şi în ce ordine
- Atributele posibile şi valorile lor prestabilite sau constante
- Valorile caracter ale tipurilor enumerate
- Entităţile neanalizate utilizate în document şi modul în care sunt referite prin nume
- Codificările de limbă utilizate în document
- Entităţile caracter utilizate în document
- Alte informaţii importante pentru prelucrarea şi redarea documentului

Respectând aceste reguli veţi fi capabil să creaţi documente în concordanţă cu şablonul pe care proiectantul documentului l-a avut în minte atunci când a creat DTD-ul. Într-un mediu care nu validează puteţi crea etichete şi atribute pe măsură ce înaintaţi.

Foaia sau manualul de codare afişează toate acestea într-un format uşor de citit şi înţeles, *dacă* autorul DTD-ului şi-a făcut treaba corect. Când creaţi un document XML sau corectaţi o eroare, este posibil să nu beneficiaţi de avantajul unui mediu de creare complet. Poate că utilizaţi vi prin telnet de la o distanţă de mii de kilometri. Este întotdeauna important să păstraţi la îndemână documentaţia de codare pentru cazul în care sunteţi apelat în miez de noapte şi sunteţi rugat să reparaţi sistemul de acces la baza dumneavoastră de date, în valoare de un milion de dolari.

Figura 2.3 din acest capitol prezintă un mediu de creare care validează; aceasta vă poate economisi mult timp şi vă poate uşura viaţa încapsulând automat informaţia DTD-ului care, altfel, este posibil să fie prezentată într-un document de codare într-o formă programatică. Totuşi, astfel de instrumente nu sunt infailibile şi nici nu sunt disponibile pentru fiecare platformă. Cu toate că ele pot genera mesaje de eroare „utile" atunci când se opresc din prelucrarea fişierului, mesajul efectiv poate să nu aibă nimic în comun cu eroarea făcută de dumneavoastră. În cele mai multe cazuri va trebui să vă bazaţi pe capul dumneavoastră, şi nu pe un instrument.

Reguli de denumire a entităţilor

Numele entităţilor trebuie să înceapă cu un caracter, care este orice caracter din cadrul unui sistem de scriere dat care corespunde conceptului general de caracter Latin, plus caracterele ASCII liniuţă de subliniere şi două puncte. Primul caracter poate fi urmat de caractere, cifre şi o selecţie de accente şi caractere de extindere reprezentând combinaţii de simboluri de un tip sau altul. Instrucţiunile de prelucrare nu pot începe cu literele ASCII XML, xml sau orice combinaţie de majuscule şi minuscule a acestor trei caractere. Utilizarea caracterului ASCII două puncte este o idee foarte rea, excepţie făcând cazul în care folosiţi zone de nume, cu toate că este posibil să faceţi aceasta în mod legal, din moment ce, într-un nume, prezenţa unui şir urmat de două puncte *arată* ca şi când s-ar referi la o zonă de nume, chiar dacă n-o face.

Notă În mod surprinzător, multe explicaţii referitoare la XML greşesc în această privinţă. În ciuda faptului că toată lumea este de acord că în conţinut puteţi utiliza codificări, faptul că această libertate se extinde şi la marcaje este mai greu de înţeles. Toată lumea este atât de obişnuită cu limitările HTML încât este dificil să îşi amintească să adopte această libertate.

Singurul loc unde *ceva* ASCII este obligatoriu este într-un DTD, care este un document din culise pe care autorii obișnuiți este posibil să nu îl vadă niciodată. Literele care formează numele identificatorilor publici – caracterele care completează partea literală a declarației `<!DOCTYPE PUBLIC "-//identificator public" "uri">` – trebuie să fie caractere ASCII deoarece acesta este un standard Internet. Referințele la caractere numerice, cum ar fi `
`, trebuie introduse utilizând un set de numere ASCII foarte restrâns. Codurile pentru limbi ISO 639 și codurile pentru țări ISO 3166 utilizate în `XML:lang` și atributele XHTML `lang` trebuie și ele să fie în ASCII, din nou datorită faptului că acestea sunt standarde Internet. Mai există câteva locuri unde este nevoie de ASCII, cum ar fi ghilimelele (care trebuie să fie ghilimele ASCII), caractere de marcare speciale, cum ar fi `>`, cuvinte cheie, ca `<!ELEMENT ... >` sau `IDREF`, și așa mai departe, dar în general nu trebuie să vă faceți griji dacă utilizați caracterele echivalente din seturile de caractere ale limbii naționale, definite în standardele Unicode/ISO/IEC 10646.

În rest, XML este prietenos cu vorbitorii de alte limbi decât engleza. Chiar și semnele care în Unicode și în ISO sunt numite „caractere de combinare" sau „caractere de extindere" sunt permise atât în etichete, cât și în conținut. Acestea sunt în primul rând accentele care pot fi trecute în nume pentru a semnala motoarelor de afișare că un accent ar trebui combinat cu caracterul precedent, indiferent cum se realizează acest lucru. Deci francezii, germanii și mulți alții care utilizează un alfabet latin pot vedea etichete ortografiate corect în propria lor limbă, cu accentele și umlaut-urile potrivite, așa cum și vorbitorii de hindusă, ebraică, arabă și multe alte limbi pot să utilizeze astfel de caractere.

Atenție Cu toate că este extraordinar faptul că XML permite caractere chinezești în marcaje și în conținut, aceasta nu înseamnă că sistemul dumneavoastră suportă în realitate afișarea și introducerea caracterelor chinezești, sau a oricăror alte sisteme de scriere ideografice, silabice sau alfabetice utilizate în întreaga lume. Pentru o funcționalitate totală veți avea nevoie atât de setul de caractere cât și de o tastatură corespunzătoare, precum și de suport în sistemul de operare.

După ani de dependență eurocentrică, chiar anglocentrică, pe Web de ASCII și de ASCII extins, XML a egalizat oarecum terenul de joc pentru toți jucătorii de pe Web, incluzând nenumărați utilizatori care vor fi capabili să acceseze informațiile în limbile lor materne. Având un singur mijloc standard de comunicare în, să zicem chineză, în care există trei „standarde" principale și un număr de variante, se va îmbunătăți cu siguranță disponibilitatea motoarelor de redare și, în același timp, li se va reduce costul.

Liste și tipuri de atribute

În foaia dumneavoastră de codare XML veți găsi o listă de atribute pentru fiecare element. Dacă utilizați un editor care validează, cum ar fi XML Pro 2.0, veți avea probabil un meniu de atribute disponibile ori de câte ori veți poziționa cursorul pe un element în modul de vizualizare a arborelui documentului. De asemenea, aveți și un meniu de elemente care pot fi inserate în fișier în acel loc.

Un astfel de editor vă poate economisi o cantitate imensă de timp deoarece elimină o parte din povara învățării unei foi sau a unui manual de codare. Totuși, o foaie de codare bună poate oferi o privire generală mult mai accesibilă și poate explica raționamentul din spatele structurii documentului, lucru pe care nici măcar un editor care validează nu îl poate face.

Figura 2.3 prezintă câteva dintre punctele tari ale unui astfel de editor. Structura arborescentă a documentului este afișată clar, cu toate că un document mare sau complex poate depăși rapid, pe un ecran mic, limita de lizibilitate. Cursorul scos în evidență vă arată unde vă aflați în document la un moment dat. Pe ecran mai este vizibilă și o listă de atribute disponibile pentru elementul evidențiat. Puteți derula lista de atribute și să le alegeți pe cele pe care doriți să le folosiți pentru acel element.

La nivelul procesorului poate fi făcută numai o verificare foarte sumară a valorilor atributelor dumneavoastră. XML în sine nu oferă facilități de validare a atributelor și a conținutului elementelor în detaliu, în afară de faptul că verifică dacă opțiunile enumerate au fost sau nu introduse corect, lucru făcut automat de un editor care validează, precum și alte câteva detalii minore. XML Pro și alte medii de creare vă pot oferi o listă autoderulantă cu valorile enumerate, pentru a evita să faceți o greșeală, dar cam asta e tot. Soluția pentru această dilemă sunt schemele XML, despre care veți învăța în Capitolul 7.

Chiar și sumara verificare a câmpurilor poate fi modestă în utilizarea efectivă. Dacă vă gândiți, de exemplu, că numerele de telefon trebuie să fie valori numerice, atunci vă aflați în fața diverselor convenții dependente de limbă care, pentru reprezentarea numerelor, pot folosi puncte, cratime, virgule, slash-uri, paranteze sau chiar cuvinte pronunțate literă cu literă. Pentru unele scopuri, armata S.U.A. utilizează o tastatură de telefon hexazecimală care la obișnuitele 0-9, * și # adaugă „cifrele" A, B, C și D. Standardele Telephone TouchTone™, numite oficial tonuri DTMF (Dual Tone Multi-Frequency), permit acest lucru, deci puteți face și dumneavoastră asta. Codurile poștale din multe țări conțin atât litere cât și cifre. Și numerele mari, precum și fracțiile zecimale, pot avea semne de punctuație în interiorul lor care să varieze de la o țară la alta, deseori o virgulă, respectiv un punct, dar destul de des un punct și o virgulă, exact cu semnificație inversă. În general, totul este mult mai complicat decât ați crede, iar adăugarea de restricții necugetate pentru intrări va cauza mai mult ca sigur probleme și va afecta imaginea dumneavoastră.

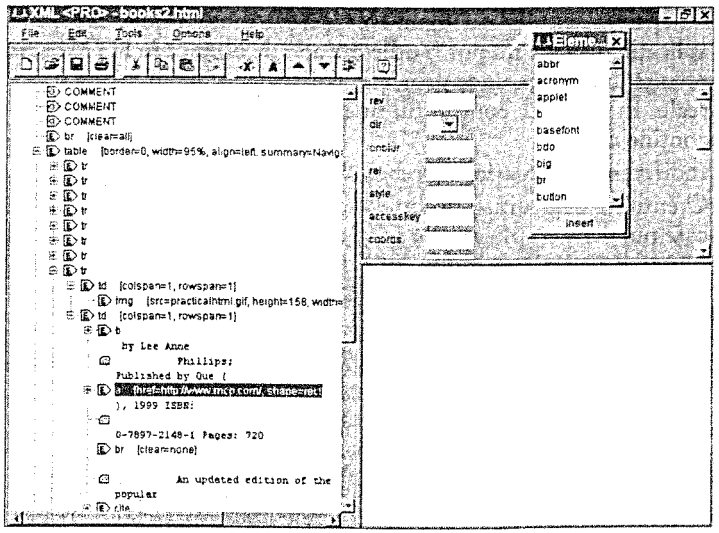


FIGURA 2.3
XML Pro 2.0 este utilizat pentru editarea unui fișier XHTML. Observați lista de atribute disponibile pentru un element ancoră, în funcție de poziția cursorului.

Atenție Într-un mediu global, merită să fiți foarte atent la ceea ce este permis și ce nu în oricare câmp de date. Cu toate că încep să apară multe scheme care permit detalii fine în constrângerile referitoare la câmpuri, exemplele date ar trebui să vă pună pe gânduri. Numerele de telefon pot arăta astfel: 011 41 09 23 44 23 în Franța sau 011 8524 99 99 22 33 într-o insulă mică din Pacific. Trebuie să găsiți un compromis adecvat între ceea ce dorește utilizatorul și ceea sunteți pregătit să acceptați. În multe situații, capacitatea de a nu ține cont de verificarea intervalului la cererea utilizatorului poate ajuta la reducerea frustrării. Clienților americani le puteți reaminti numărul „corect" de cifre dintr-un număr de telefon, în timp ce le permiteți celorlalți, dacă insistă, să introducă orice doresc.

Cu toate că o aplicație XML poate impune constrângeri numai într-o bază de date subiacentă, înainte de finalizarea analizei dumneavoastră gândiți-vă că este posibil ca baza de date să fi fost proiectată într-o manieră insulară sau limitată și să nu îndeplinească cerințele reale ale unei piețe globale. Unele date au în mod firesc o natură regională, iar o persoană matură este conștientă de acest fapt. Dacă o persoană mănâncă ținând furculița în mâna dreaptă sau în mâna stângă, este o chestie de gust și nu cere verificarea erorilor. În cazul diferențelor cosmetice, gândiți-vă la normalizarea datelor înainte de verificarea erorilor sau de stocare. Înainte ca intrarea sa să fie acceptată, nu există un motiv rezonabil pentru care un utilizator britanic ar trebui să ghicească faptul că, deseori, în SUA sau în Canada prefixele telefonice se încadrează între paranteze.

Entități neanalizate

O entitate neanalizată este orice lucru care nu poate fi recunoscut de un procesor XML, fie date binare, cum ar fi un fișier imagine sau audio, fie text care trebuie să fie transferat unei aplicații fără a fi prelucrat în vreun fel. HTML utilizează comentarii pentru a ascunde un astfel de text de browserul HTML, dar XML are câteva mecanisme care funcționează mult mai sigur. De fapt, XML nu trebuie să transfere deloc comentarii unei aplicații, deci ele nu pot fi utilizate atât de liber ca în HTML. Aceasta a fost, cred, o mică revoltă din partea unor proiectanți care detestau ideea de a utiliza comentarii pentru date reale. În HTML, conținutul unei etichete `<SCRIPT>` este definit să fie `PCDATA`, care poate conține orice.

Indiferent de situație, ar trebui să utilizați noile reguli ale XML.

O entitate neanalizată trebuie mai întâi să fie declarată ca `NOTATION`, o declarație specială care numește o aplicație de asistență care știe cum să lucreze cu entități de un anumit tip. Notației îi dați un nume, un identificator public opțional și apoi numele mai puțin opțional al aplicației de asistență, ca în una din variantele următoare:

```
<!NOTATION nume-mnemonic PUBLIC "identificator-public">
<!NOTATION nume-mnemonic PUBLIC "identificator-public" "nume-aplicatie.exe " >
<!NOTATION nume-mnemonic SYSTEM "nume-aplicatie.exe " >
```

Prima opțiune funcționează numai dacă aveți un catalog. Cea de a doua și cea de a treia funcționează indiferent dacă aveți sau nu un catalog. Nu vă puteți baza pe un catalog, deoarece acesta este un instrument SGML pe care multe procesoare XML actuale l-au moștenit implicit de la predecesoarele lor SGML. Studierea catalogului nu este specificată în recomandarea W3C și nu se poate conta niciodată pe ea. Dacă este posibil, utilizați ultimele două versiuni. Pe de altă parte, codarea hard a informațiilor despre locația

și identitatea aplicației de asistență în absolut fiecare DTD este un anacronism predispus la erori pe Web.

Atenție Prin redefinirea comportamentului script-urilor în prezența comentariilor, proiectanții XML-ului au introdus o problemă de incompatibilitate între XML și HTML. După toate probabilitățile, procesoarele XML vor continua să transfere comentarii la aplicații deoarece multe pagini se vor întrerupe dacă nu vor avea acest comportament. De asemenea, procesoarelor le este *permis* transferul informației comentate prin același limbaj prin care li se *permite* să nu o facă.

După ce ați definit o notație a unei entități neanalizate, ea trebuie să fie declarată ca o entitate, astfel:

```
<!ENTITY nume-mnemonic NDATA nume-mnemonic>
```

și apoi trecută ca un atribut într-un element pentru a o putea utiliza:

```
<!ELEMENT name EMPTY>
<!ATTLIST name type NOTATION "numemnemonic"
        … >
```

Numele mnemonice nu împart aceeași zonă de nume, deci nu are importanță dacă ele sunt identice, ceea ce eu încurajez numai de dragul clarității.

În acest moment vă întoarceți la foaia de codare și puteți utiliza tipul de date definit astfel. Tipul de date poate fi utilizat numai ca atribut al unui element declarat ca fiind de acel tip sau având acel tip disponibil într-o enumerare. Celelalte atribute adună informațiile de care are nevoie aplicația de asistență externă pentru a fi capabilă să prelucreze datele.

O aplicație tipică poate fi un fișier imagine:

```
<image source="uri" alt="graphic description" type="gif89a">
```

Acest element va necesita în DTD următoarele declarații:

```
<!NOTATION gif89a PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
➥ 89a//EN" "explorer.exe">
<!ENTITY gif89a NDATA gif89a>
<!ELEMENT image EMPTY>
<!ATTLIST image source CDATA #REQUIRED
                alt   CDATA #IMPLIED
                type  NDATA gif89a >
```

Pentru majoritatea instrumentelor, nu va avea importanță dacă formatul îl specificați ca `gif87a` sau ca `gif89a`, deoarece aceleași instrumente manevrează ambele formate. Este oricum lipsit de logică să așteptați ca persoana care codifică fișierul să cunoască diferența dintre cele două.

Notațiile vor fi mult îmbunătățite o dată cu adăugarea facilităților oferite de XLink/XPointer care ajută la urmărirea locațiilor asistenților. Cu instabilitatea generală a Web-ului la nivelul său actual și cu larga varietate de facilități și arhitecturi de pe sistemele utilizatorilor, orice ajutor care poate fi obținut de bietul utilizator va face mai ușoară configurarea instrumentelor XML. DTD-ul, în forma sa actuală, necesită mult prea multă ajustare în stil UNIX a fișierelor pentru a fi pe placul utilizatorilor obișnuiți.

Notă XLink şi XPointer încearcă să învingă limitările tehnologiei pointer actuale. Permit toate tipurile de relaţii, inclusiv pointeri inverşi care să fie generaţi pe loc în documente fără acces la scriere, realizându-şi magia chiar asupra copiei afişate, şi nu prin inserarea fizică brută a etichetelor în conţinut.

Cerinţele XML pentru notaţii, în forma actuală, sunt extrem de severe. Posibilitatea de a indica locaţia unde se află aplicaţia de asistenţă poate fi comodă pentru notaţii neobişnuite sau foarte specializate. Totuşi, este de aşteptat ca agentul utilizator să ştie cum să afişeze multe dintre tipurile mai obişnuite, cum ar fi GIF-uri, JPEG-uri, PNG-uri, WAV-uri şi alte tipuri de fişiere binare mai mult sau mai puţin standard, utilizate pe Web. Iată o listă a notaţiilor obişnuite:

```
<!NOTATION eps        PUBLIC "+//ISBN 0-201-18127-4::Adobe//NOTATION
  ➥Postscript Language Reference Manual//EN" >
<!NOTATION tex        PUBLIC "+//ISBN 0-201-13448-9::Knuth//NOTATION The
  ➥TeXbook//EN" >
<!NOTATION cgmchar    PUBLIC "ISO 8632/2//NOTATION Character encoding//EN" >
<!NOTATION cgmbinary  PUBLIC "ISO 8632/3//NOTATION Binary encoding//EN" >
<!NOTATION cgmclear   PUBLIC "ISO 8632/4//NOTATION Clear text encoding//EN" >
<!NOTATION tiff       PUBLIC "ISO 12083:1994//NOTATION TIFF-1//EN" >
<!NOTATION jpeg       PUBLIC "ISO/IEC 10918:1983//NOTATION Digital
  ➥Compression and Encoding of Continuous-tone Still Images (JPEG)//EN" >
<!NOTATION gif87a     PUBLIC "-//CompuServe//NOTATION Graphics Interchange
  ➥Format 87a//EN" >
<!NOTATION gif89a     PUBLIC "-//CompuServe//NOTATION Graphics Interchange
  ➥Format 89a//EN" >
<!NOTATION fax        PUBLIC "-//USA-DOD//NOTATION CCITT Group 4 Facsimile
  ➥Type 1 Untiled Raster//EN" >
```

Desigur, va trebui să adăugaţi un ID sistem pentru a le putea utiliza, fie ca pointer la o aplicaţie de asistenţă locală, fie în fişierul catalog care centralizează locaţiile acestor aplicaţii, dacă este disponibil pentru instrumentele dumneavoastră.

Aplicaţii din lumea reală

Aţi văzut deja, mai sus, unele dintre instrumentele pe care este posibil să le utilizaţi pentru a crea un document XML, dar unde găsiţi DTD-urile după care veţi crea? Ei bine, puteţi să citiţi această carte în continuare şi să scrieţi propriul dumneavoastră DTD sau puteţi utiliza unul deja existent.

Multe DTD-uri se află în domenii publice sau sunt disponibile ca standarde de la ISO, ANSI sau de la alte organisme de standardizare. La sfârşitul acestei cărţi există o listă cuprinzătoare, dar aici voi menţiona câteva dintre aplicaţiile mai importante ale XML care se afirmă în lumea de azi:

- Health Level-7 (HL7), Standardul informatic de sănătate (Health Informatics Standard) a fost introdus în anul 1987 pentru a dezvolta standarde pentru schimbul electronic de informaţii clinice, financiare şi administrative între sistemele de calcul din domeniul sănătăţii. HL7 s-a concentrat asupra utilizării SGML şi XML ca mecanisme de transport între diferite sisteme informatice din domeniul sănătăţii.

- Real Estate Transaction Standard (RETS – standardul de tranzacţii imobiliare) este o metodă bazată pe XML pentru schimbul de informaţii referitoare la tranzacţiile imobiliare. Un standard concurent este Real Estate Markup Language (RELML – limbajul de marcare pentru domeniul imobiliar) care utilizează DTD-uri XML pentru a prezenta listinguri cu spaţii pentru locuinţe, spaţii comerciale şi terenuri virane, pentru publicarea lor pe Web.
- RosettaNet, limbajul comun pentru afaceri, este o iniţiativă EDI/E-Commerce destinată intermedierilor în domeniul calculatoarelor.
- MathML şi CML sunt două standarde XML ştiinţifice care permit matematicienilor să editeze ecuaţii, iar chimiştilor să prezinte formule chimice.
- SMIL, the Synchronized Multimedia Integration Language (limbajul de integrare multimedia sincronizat), este HyTime for Everyman (HyTime pentru fiecare), un limbaj de marcare multimedia care permite furnizorilor de conţinut să realizeze prezentări audio şi video complexe.
- ICE, Information and Content Exchange (schimb de informaţii şi conţinut), cu toate că nu este chiar o aplicaţie XML, fiind de fapt un mecanism de transport, permite schimbul de investiţii on-line şi de informaţii personale pe Web.
- SAE J2008 este un sistem de comenzi şi inventariere bazat pe XML pentru industria auto; MISTI, the Missile Industry Supply-chain Transaction Infrastructures, face acelaşi lucru pentru industria spaţială.
- Chinese DTDs furnizează structura specializată necesară pentru publicarea în limba chineză. DTD-uri similare există pentru japoneză, coreeană, vietnameză şi multe alte limbi.
- GedML, un standard XML genealogic, încurajează fluxul liber pe Web al datelor genealogice. Există deja programe pentru conversia fişierelor standard GEnealogical Data COMmunication (GEDCOM) la GedML.

Iar lista poate continua. După cum puteţi vedea, gama aplicaţiilor este imensă, atingând aproape fiecare zonă a preocupărilor umane. Puţine firme pot ignora liniştite XML, cu toate că sunt atât de multe standarde existente şi propuse în atât de multe domenii, încât în mod sigur unele firme vor fi eliminate, în timp ce competitorii importanţi se străduiesc să ajungă în vârf printr-un duel zgomotos de propuneri.

Sugestie de la LeeAnne.com Words to weave by

Cu atât de multe propuneri patentate puteţi să vă întrebaţi care sunt diferenţele între ele. De obicei, va trebui să cereţi furnizorilor potenţiali să îşi dezvăluie DTD-ul, deoarece sunt puţine DTD-uri disponibile pe Web. Există două depozite de unde puteţi începe. Pentru platformele Microsoft, consorţiul BizTalk, la `http://www.biztalk.org/`, dispune de o listă în care puteţi căuta, deşi aceasta nu este uşor de folosit şi vă cere să ghiciţi care sunt cuvintele cheie potrivite pentru descrierea tipului de DTD pe care îl căutaţi. Organizaţia pentru progresul standardelor de informaţie structurată (OASIS – the Organisation for the Advancement of Structured Information Standards), la `http://www.oasis-open.org/`, plănuieşte un alt depozit, dar deocamdată oferă pentru accesul public doar DTD-ul DocBook şi o submulţime a DTD-ului pentru tabele CALS, numită modelul de tabel pentru schimburi (Exchange Table Model). Mulţi furnizori tratează propriile DTD-uri de parcă ar fi secrete de stat.

Fiecare producător de browsere are standarde propuse, despre care ceilalți se plâng că sunt îndreptate împotriva lor. Așa cum războaiele browserelor au dus la dezvoltarea unor „extensii" patentate la HTML, care tindeau (sau încercau) să blocheze celelalte browsere, creând un turn Babel de metode incompatibile care ne chinuie și astăzi, XML va fi într-o continuă schimbare pentru o bună perioadă de timp.

Totuși, bazele există deja, iar o comunitate de utilizatori cerând din ce în ce mai insistent standarde deschise conduce diferitele propuneri spre convergență. Multe din succesele majore au fost standarde ale ISO și ANSI, care vând documentația pentru a-și finanța eforturile în crearea de standarde, dar asigură un teren neutru pentru toți partenerii. Pentru prețul documentației, de obicei câteva sute de dolari, toți pot concura la același nivel.

Studii de caz

Acum, când știți câte ceva despre fundamentele XML, trebuie să decideți cum să îl utilizați. Ați învățat deja că DTD-urile reprezintă o metodă de descriere precisă a convențiilor referitoare la formatul și macheta unui document XML și de ce ar trebui utilizate în unele situații și evitate în altele. Ați învățat, de asemenea, despre unele diferențe dintre XML și XHTML și despre faptul că introduc o mare diferență pentru întregul Web prin internaționalizarea seturilor de caractere și posibilitatea de utilizare a diferitelor limbi în etichete. În capitolele următoare veți învăța despre unele dintre facilitățile încorporate în XML privind securitatea și confidențialitatea, care îl fac potrivit pentru comunicațiile de afaceri B2B – business to business.

Unele dintre aceste facilități de securitate au fost deja introduse într-un hardware care implementează prelucrarea tranzacțiilor, semnături digitale și Secure Socket Layer (SSL) pe siliciu, echivalentul XML al unui cip accelerator grafic. Aceasta înseamnă că infrastructura necesară pentru suportul prelucrării tranzacțiilor XML de volum mare devine într-adevăr disponibilă.

Definirea unui sistem de prelucrare a tranzacțiilor B2B necesită abilitatea de a comunica foarte clar aspectul înregistrărilor care vor fi transferate între sisteme, deoarece acele sisteme pot fi pe platforme diferite sau pot fi implementate în limbaje de programare diferite. Un DTD XML este ideal pentru comunicarea acestor informații, deoarece poate fi garantat faptul că nu este ambiguu și că este independent de platformă. După ce tranzacția este implementată într-o aplicație, nici unul dintre sistemele implicate într-o tranzacție nu va avea nevoie să acceseze DTD-ul, deoarece toate cunoștințele despre DTD necesare pentru orice cerere de prelucrare vor fi captate în codul programului care manevrează tranzacția la fiecare capăt.

Acest scenariu este general pentru orice aplicație de comunicare între procese care utilizează XML; etapele inițiale necesită acorduri mutuale asupra DTD-ului care servește ca parte a documentației sistemului și ca facilitate de testare, dar nevoia pentru un DTD dispare după ce sistemul este gata și rulează; aceasta spre deosebire de documentele care pot servi oricui fără un proces prestabilit și pregătit pentru tranzacții aflat la celălalt capăt. Atunci când un document este vizualizat cu un browser obișnuit, informația conținută în DTD poate fi necesară pentru prezentarea corectă și manevrarea informației conținute acolo. Deci DTD-urile sunt în mod potențial mai importante pentru utilizatorii ocazionali decât pentru comunicarea între parteneri care se află într-o relație neîntreruptă.

Definițiile tipurilor documentului XML

CAPITOLUL 3

În acest capitol

- Analiza definiției tipului documentului XML
- Utilizarea notației și sintaxei definiției tipului documentului
- Valorile literale
- Declarația `NOTATION`
- Crearea declarațiilor `ATTLIST`
- Utilizarea construcțiilor de tipuri de date XML speciale
- Diferența dintre un XML construit corect și un XML valid
- Modul de utilizare a DTD-urilor externe și a fragmentelor DTD
- Modificarea unui DTD XML
- Studii de caz

Analiza definiției tipului documentului XML

Un DTD (Document Type Definition – definiția tipului documentului) definește forma și sintaxa construcțiilor din limbajul XML. DTD-urile le creați utilizând reguli simple, care pot părea mult mai complexe decât sunt în realitate. Creând sau modificând un DTD și făcându-l disponibil pe Web, faceți ca un limbaj nou sau extins să fie inteligibil și utilizabil de către oricine are un browser care suportă XML. Majoritatea paginilor XML utilizează DTD-uri publice, ceea ce le permite să schimbe date cu alți utilizatori, într-un format standard. Extensiile locale la un DTD standard permit utilizarea extensiilor structurii proprii, care nu sunt relevante pentru alți utilizatori.

Un DTD nu este necesar pentru a realiza un document XML construit corect. De fapt, în cazul în care codul dumneavoastră XML a fost generat automat, puteți să optați să săriți peste acest pas. Totuși, pentru a profita de posibilitatea de a verifica validitatea documentului dumneavoastră XML, veți avea nevoie de un DTD. Și, pentru a partaja cu ușurință datele dumneavoastră pe Web, un DTD este un pas necesar spre universalitate. Un DTD vă permite să captați și să validați posibilitățile complete ale structurii dumneavoastră de date în locul unor improvizări spontane atunci când doriți să treceți într-un domeniu opțional.

Într-un document complex, orice document particular poate să utilizeze doar un subset al posibilităților complete definite de designul documentului. Prin identificarea părților opționale dintr-un DTD, păstrați controlul asupra locațiilor lor din structură chiar dacă nu sunt utilizate în fiecare situație. În același timp, cu ajutorul unui DTD vă asigurați că părțile obligatorii sunt întotdeauna acolo, chiar dacă documentul este construit corect și fără el. Un DTD vă permite să partajați structura documentului dumneavoastră și informația întruchipată de acea structură cu o audiență mult mai mare decât era posibil înainte.

Nu trebuie să utilizați formate de date patentate care cer ambelor părți să dispună de același software, o cheltuială enormă dacă lucrați cu mulți colaboratori, clienți sau furnizori. În schimb, puteți publica structura datelor dumneavoastră acolo unde poate fi vizualizată și utilizată gratuit, sau puteți schimba un fișier cu fiecare furnizor/client, ceea ce furnizează o cheie pentru înțelegerea modului în care fiecare câmp se leagă la documentul ca întreg.

Notă Un DTD este o descriere precisă a structurii de date care poate fi transferată unui instrument automatizat. Aceasta mărește posibilitatea automatizării conversiei datelor dintr-un format în altul, sau dintr-un limbaj în altul, cu o intervenție umană minimă.

XML a fost proiectat într-un mod care permite procesoarelor XML și aplicațiilor să vizualizeze și să utilizeze documente XML fără a accesa un DTD, chiar dacă acesta există. Dacă o legătură se întrerupe, la urma urmelor, nu doriți ca datele din document să devină inutile. Deci un DTD este doar o parte a imaginii de ansamblu – o parte importantă, însă doar o parte.

Utilizarea notației și sintaxei definiției tipului documentului

În general, majoritatea DTD-urilor profesionale grupează declarațiile lor după tip și într-o ordine logică, ceea ce le face mult mai ușor de citit și de înțeles. Ordinea exactă este critică în unele cazuri, dar nu și în altele, după cum informația este necesară sau nu la momentul în care este realizată o intrare într-un tabel de căutare. Unele extinderi de entități se realizează doar în momentul utilizării. Entitățile parametru sunt necesare aproape întotdeauna, deci ar trebui, de obicei, să apară primele.

Sugestie de la LeeAnne.com Words to weave by

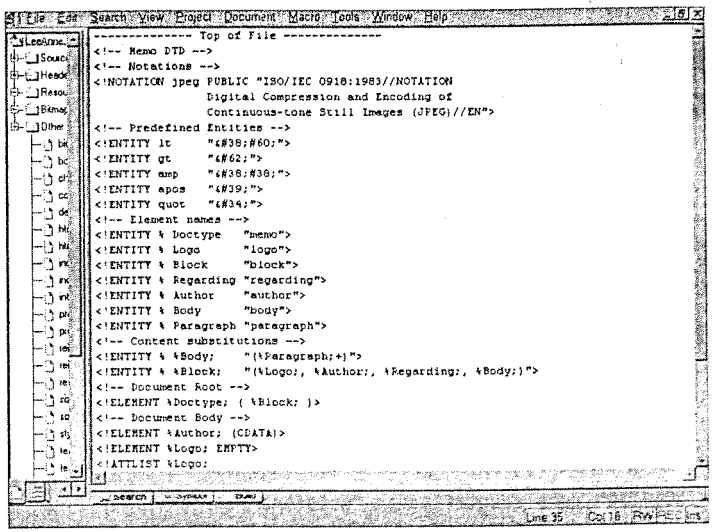
Pentru a evita grijile privind lucrurile care sunt necesare și care nu, o deprindere bună este declararea oricărui lucru înainte de utilizarea sa. Aceasta ușurează puțin și găsirea lucrurilor deoarece, de obicei, puteți căuta spre începutul DTD-ului pentru a găsi o declarație anterioară.

Ar trebui să faceți ca rădăcina arborelui documentului dumneavoastră să fie primul ELEMENT, deoarece multe procesoare presupun că primul element este chiar rădăcina. Aceasta oglindește într-o oarecare măsură structura unui XML construit corect, deoarece prima etichetă întâlnită este întotdeauna rădăcina.

Ca o regulă generală și pentru ușurarea utilizării, declarațiile ENTITY ar trebui făcute înainte de declarațiile ELEMENT și ATTLIST care le utilizează. Declarațiile ELEMENT trebuie

realizate înainte de declarațiile ATTLIST care le modifică. Declarațiile NOTATION trebuie trecute înaintea oricărei utilizări a notației respective. Figura 3.1 prezintă un exemplu de codare corectă XML.

FIGURA 3.1
Acesta este un exemplu scurt de cod DTD XML bine organizat.



Observați că fiecare secțiune afișează diferitele declarații DTD într-un grup logic, fiecare ATTLIST urmând imediat după acel ELEMENT pe care îl modifică, iar restul este aranjat după tip. Aceasta ușurează găsirea fiecărui fragment atunci când îl căutați.

Valorile literale

O *valoare literală* este orice șir încadrat între semne de citat, care nu conține semnul de citat utilizat ca delimitator pentru acel șir. Aceste marcaje pot fi fie ghilimele, fie apostrofuri, după dorința celui care scrie codul. Dacă doriți să faceți o valoare literală dintr-un șir care conține atât apostrofuri cât și ghilimele, puteți să le includeți în secvențe escape pe cele cu care doriți să delimitați șirul. Valorile literale sunt utilizate pentru specificarea conținutului unei entități interne, a valorilor atributelor sau a identificatorilor externi. În continuare sunt prezentate câteva exemple de valori literale:

- Valori literale: `"Sprintena vulpe maro sari peste cainele lenes."`
- Forma alternativă: `'Gaina lenesa cotcodaceste "fericita" din cuibarul dezordonat'`
- Forma cu secvențe escape: `'Acum este timpul pentru citatul lui taticu' "socru mare":'`

Primul exemplu utilizează ghilimele, ca în Statele Unite. Cel de-al doilea utilizează apostrofuri, ca în Marea Britanie și în multe alte locuri din lume unde se vorbește engleza. Cel de-al treilea exemplu conține atât apostrof cât și ghilimele, deci apostroful este introdus într-o secvență escape utilizându-se o entitate caracter.

Declaraţia NOTATION

O *notaţie* este orice lucru care nu poate fi înţeles şi analizat de un procesor XML (notaţiile se mai numesc şi *entităţi neanalizate*). Cu toate că aceasta evocă ideea de date binare, notaţia poate fi şi un text care nu este înţeles de XML. Un fragment de JavaScript, de exemplu, poate fi păstrat într-un fişier extern şi să fie referit ca o notaţie.

Atenţie Notaţiile pot referi doar fişiere externe. Nu există nici o metodă de a ascunde informaţii în interiorul unui document şi a le transfera unui agent utilizator pentru o prelucrare specială, decât dacă le puneţi în interiorul unei etichete XML şi instruiţi agentul să facă ceva cu conţinutul etichetei. Dacă doriţi să includeţi în document date de tip text conţinând caractere speciale, ar trebui să le treceţi în interiorul unui element CDATA, după cum este descris în secţiunea „Trecerea unui bloc de text într-o secţiune CDATA".

Problema cu notaţiile este că ele necesită acces la DTD pentru a putea fi utilizate. În ciuda faptului că puteţi utiliza submulţimea internă a DTD-ului pentru a face disponibilă local în document informaţia notaţiei, analizoarelor care nu validează nu li se cere să citească submulţimea internă a DTD-ului.

Sintaxa NOTATION este simplă:

```
<!NOTATION nume identificator "asistent" >
```

Identificatorul este o intrare din catalog, aşa cum se utilizează în SGML. Multe procesoare XML sunt procesoare SGML reciclate, deci suportă un catalog în mod prestabilit. Este un pic mai sigur decât indicarea spre o aplicaţie de asistenţă care este posibil să nu se găsească în acel loc, dar XML cere ca aplicaţia de asistenţă să fie totuşi referită, fapt care poate duce la comportamente contradictorii. Puteţi să referiţi Adobe Photoshop, de exemplu, ca aplicaţie de asistenţă pentru vizualizarea unei imagini GIF, dar este foarte probabil că browserul ştie şi singur să afişeze GIF-uri. De asemenea, este mult mai probabil că browserul este capabil să integreze corect imaginea în documentul redat pe un dispozitiv de afişare sau la imprimantă, o operaţie pe care Adobe Photoshop nu prea poate să o facă. Utilizarea atât a identificatorului cât şi a „numelui" asistentului vă permite să faceţi un compromis între comunicarea către agentul utilizator a tipului de fişier care va fi transferat şi comunicarea explicită a modului de afişare a fişierului, neştiind nimic despre mediul în care va fi afişat documentul.

Cu toate că unele persoane vă pot încuraja să vă comportaţi ca şi când Microsoft Windows ar fi centrul universului şi să scrieţi ceva de genul

```
<!NOTATION gif SYSTEM "gif">
```

care, într-adevăr, funcţionează pe sistemele Windows, eu nu sunt una dintre ele. Utilizaţi această sintaxă pe riscul dumneavoastră. Exemplul de mai sus presupune existenţa lui Registry pe sistemele Windows care rezolvă referinţa, registru care nu este chiar universal valabil şi scurtcircuitează complet sistemul standard de identificare.

O conduită mai sigură este să introduceţi atât întreaga intrare din catalog, cât şi secvenţa identificatorului. Această metodă oferă o sugestie mai puternică, despre ce poate fi făcut cu acest fişier, aplicaţiei care va lucra cu fişierul XML prelucrat, dar nu prelucrează efectiv referinţa notaţiei. Identificatorul sistem "gif" funcţionează pe sistemele Windows, deoarece Windows cunoaşte deja GIF-urile. Dar un dispozitiv portabil sau chiar un calculator care utilizează un alt sistem de operare este posibil să nu aibă la îndemână informaţii despre manevrarea imaginilor GIF.

Deci ar fi mai bine să refaceţi referinţa prezentată anterior, astfel:

```
<!NOTATION gif89a PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
➥89a//EN" "gif" >
```

Notaţiile nu pot fi folosite izolat. Ele trebuie declarate şi într-o entitate. O secvenţă de declaraţii completă poate arăta astfel:

```
<!NOTATION gif89a    PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
➥89a//EN" "gif">
<!ENTITY gif89a SYSTEM "gif89a.gif" NDATA gif89a>
<!ELEMENT image EMPTY>
<!ATTLIST image source CDATA #REQUIRED
                alt    CDATA #IMPLIED
                type   NDATA gif89a >
```

În documentul dumneavoastră, eticheta va arăta astfel:

```
<image source="uri"
       alt=[o imagine]>
```

Puteţi, de asemenea, să creaţi o listă enumerată de tipuri de notaţii, care utilizează o sintaxă uşor diferită pentru a descrie tipul notaţiei:

```
<!NOTATION gif87a    PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
➥87a//EN" "gif">
<!NOTATION gif89a    PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
➥89a//EN" "gif">
<!ENTITY gif87a SYSTEM "gif87a.gif" NDATA gif87a>
<!ENTITY gif89a SYSTEM "gif89a.gif" NDATA gif89a>
<!ELEMENT image EMPTY>
<!ATTLIST image source CDATA #REQUIRED
                alt    CDATA #IMPLIED
                type   NOTATION (gif87a | gif89a) "gif89a" >
```

În documentul dumneavoastră, eticheta va arăta astfel:

```
<image source="uri"
       alt=[o imagine]>
```

dar puteţi să nu ţineţi cont de tipul prestabilit oferit în lista de atribute, astfel:

```
<image source="uri"
       alt=[o imagine]
       type="gif87a" >
```

Aceasta permite agentului utilizator să ştie că imaginea este în formatul GIF87a, dacă aceasta are vreo importanţă.

Notaţiile sunt soluţii improprii

Notaţiile sunt mecanisme interesante deoarece vă permit să izolaţi atât datele binare, cât şi datele de tip caracter, cum ar fi script-uri sau coduri sursă interpretate cu care nu doriţi să aibă de lucru procesorul XML. Din acest punct de vedere, ele sunt un lucru bun. Dar specificitatea necesară devine repede obositoare. XML este utilizat aici ca un dispecer care trebuie să aleagă între alternative care se învechesc repede, pe măsura apariţiei noilor tehnologii binare.

Sintaxa lui NOTATION este un fel de anacronism rămas de la SGML, care o utilizează adesea. Este posibil să fi fost inclusă pentru compatibilitate cu un standard mai vechi, dar pe Web este aproape inimaginabil că puteți ști locația *permanentă* a oricărui lucru, chiar și pe propriul dumneavoastră sistem. Sistemele de fișiere evoluează și se modifică; nimic nu este constant. Declarațiile NOTATION presupun implicit un mediu care rămâne neschimbat sau care se modifică atât de încet încât ajustarea unei intrări din catalog nu este o sarcină dificilă.

Web-ul este diferit. Modificările se propagă peste noapte. Înainte să poți clipi, cineva are acolo un nou dispozitiv multimedia și *toată lumea* îl folosește. Este posibil ca mecanismul plug-in utilizat deja de browsere să fie un mod mai bun de a face acest lucru. Sau poate că am putea numai să declarăm o notație plug-in. Ar fi chiar mai bine să lăsăm browserul și serverul să negocieze asupra celui mai bun format pentru o anumită situație. Din păcate, metoda cea mai bună urmează să apară. Deci până atunci, utilizarea metodei oficiale este o soluție interimară. Se speră că orice mecanism nou care va înlătura această soluție improprie va fi capabil să utilizeze metoda mai veche pentru a extrapola de la locația codată hard la locul în care poate fi găsit un procesor corespunzător. În orice caz, ar trebui să vă gândiți mult și bine înainte de a include notații într-un document XML și să fiți pregătit pentru ceva mai bun care, în viitorul apropiat, să ia locul notațiilor.

Declarația ENTITY

Declarația ENTITY este una dintre cele mai simple, în ciuda faptului că există atât de multe tipuri diferite de entități. Lista restricțiilor referitoare la entități este oricum confuză și este descrisă atât de succint în Recomandarea XML 1.0 încât trebuie să o citiți cu atenție o dată sau de două ori pentru a o înțelege. Veți învăța mai multe despre entități și veți avea prilejul favorabil de a compara și opune variatele lor utilizări în secțiunea „Tabelul de entități W3C" din acest capitol.

În cazul în care considerați un document XML ca fiind o colecție de entități care sunt, în linii mari, obiecte, în sensul orientării spre obiecte, declarația unei entități este o metodă de a indica o instanță a unui anumit obiect. Nu există multe opțiuni și declarația nu este foarte greu de învățat. Formatul de bază este următorul:

```
<!ENTITY nume valoare>
```

Partea nume are două opțiuni, una fără modificator, după cum vedeți aici, și una cu un semn procent (%) care precedă numele, urmat de un spațiu care marchează o entitate parametru. Partea valoare are trei opțiuni de bază: fie un șir încadrat între ghilimele, un identificator care indică o intrare din catalog sau o locație externă fișierului, fie o referință de notație. În continuare sunt prezentate câteva variante pentru entități analizate:

```
<!ENTITY nume PUBLIC "intrare catalog" >
<!ENTITY nume PUBLIC "intrare catalog" "uri" >
<!ENTITY nume SYSTEM "uri" > <!-- ID extern -->
<!ENTITY nume "&#0000;" > <!-- Entitate generala mnemonica pentru entitati caracter -->
```

În continuare este prezentat modul în care referiți entitățile parametru:

```
<!ENTITY % nume PUBLIC "intrare catalog" >
<!ENTITY % nume PUBLIC "intrare catalog" "uri" >
<!ENTITY % nume SYSTEM "uri" > <!-- ID extern -->
<!ENTITY % nume "&#0000;" > <!-- Entitate parametru mnemonica pentru entitati caracter -->
```


Entitățile parametru se comportă diferit față de entitățile generale, deoarece au fost proiectate pentru scopuri diferite. Diferența din declarațiile lor este proiectată pentru a fi evidentă atunci când dumneavoastră, sau procesorul XML, vedeți o declarație.

Modul în care le referiți este și el diferit. O entitate generală se referă astfel:

```
&nume;
```

Iar o entitate parametru se referă astfel:

```
%nume;
```

Aceasta este o diferență relativ minoră care maschează o diferență imensă la utilizare.

Entități parametru

Entitatea parametru seamănă un pic cu o macrocomandă C proiectată pentru a fi utilizată chiar de către DTD, deci puteți identifica obiecte – colecții de date – pentru utilizarea lor în timpul procesului de construire a DTD-ului. După ce acest lucru este făcut, entitățile parametru nu mai au nicăieri nici o semnificație. Deci dacă se întâmplă ca documentul dumneavoastră să conțină un șir care arată ca numele unei entități parametru, el va fi ignorat, sau mai degrabă tratat ca un text simplu, ceea ce și este.

Totuși, în interiorul DTD-ului, o entitate parametru are o mare utilitate. O puteți utiliza pentru a stoca pentru o utilizare ulterioară fragmente de text marcat sau, pentru comoditate, să indice la colecții externe de text marcat.

În submulțimea internă a DTD-ului, entitățile parametru pot fi utilizate numai pentru a insera marcaje complete. Deci, următorul exemplu de declarație și utilizare este corect:

```
<!DOCTYPE nume [
<!ENTITY % numelemeu "<!ELEMENT e1 ANY>">
%numelemeu;
[...]
]
```

Exemplul următor, utilizând referințe externe indicate printr-un URI și, în cea de a doua entitate, un ID public (sau intrare din catalog), este de asemenea corect:

```
<!DOCTYPE nume [
<!ENTITY % numelemeu SYSTEM "uri">
%numelemeu;
<!ENTITY % numelemeu2 PUBLIC "intrare din catalog" "uri">
%numelemeu;
 [...]
]
```

Totuși, acest exemplu care încearcă să definească părți de marcaj care vor fi rezolvate ulterior nu este legal în submulțimea internă a DTD-ului:

```
<!DOCTYPE nume [
<!ENTITY % parteamea "ANY">
<!ENTITY % numelemeu "<!ELEMENT e1 %parteamea;>">
%numelemeu;
[...]
]
```

Observați că acest cod *ar fi* corect în submulțimea externă a DTD-ului sau într-o entitate externă.

Într-un procesor XML care nu validează, referințele externe pot sau nu să fie preluate și încorporate în document, dar aceasta nu este o eroare, indiferent de situație.

Notă Diferența dintre procesoarele XML care validează și cele care nu validează poate părea minoră, dar aproape toate browserele sunt și vor fi dintre cele care nu validează. Pe Web, va fi posibil ca un DTD să refere zeci de locații, oricare dintre ele fiind capabilă să refere alte zeci. Un browser care validează trebuie să interpreteze *totul*, în mod potențial întregul Web, înainte să afișeze *orice*. Așteptarea poate deveni obositoare.

Acest fapt ridică o problemă subtilă. Dacă submulțimea internă a DTD-ului conține un amestec de referințe de entități parametru interne și externe, un procesor care nu validează trebuie să oprească prelucrarea lor imediat ce nu a interpretat una dintre ele, fapt permis de altfel. Motivul este că referința poate deveni nedefinită:

```
<!DOCTYPE nume [
<!ENTITY % numelemeu "<!ELEMENT element1 ANY>">
%numelemeu1;
<!ENTITY % numelemeu2 SYSTEM "uri"> <!-- fisierul extern contine <!ELEMENT element2
ANY>
%numelemeu2;
<!ENTITY % numelemeu3 "<!ELEMENT element3 ANY>">
%numelemeu3;
 [...]
]
```

Dacă procesorul XML care nu validează citește entitățile parametru externe, lucru permis, atunci toate cele trei elemente sunt declarate. Dacă nu citește nici o entitate parametru externă, lucru de asemenea permis, atunci este declarat numai `element1`. Motivul este că procesorul nu știe dacă referința externă conține sau nu o declarație a lui `element3` în textul ei, caz în care valoarea lui `element3` va fi cea care era, deoarece procesorul *va trebui* să îl vadă primul *dacă* l-a citit. Deoarece nu știe sigur, el *trebuie* să ignore toate referințele entitate și listă de atribute care urmează. Pentru a complica și mai mult lucrurile, unui procesor XML care nu validează îi este permis să sară *toate* entitățile parametru, caz în care nici un element nu este definit.

Până la acest punct, oricum, un procesor XML care nu validează trebuie să citească și să prelucreze submulțimea internă a DTD-ului, dacă o astfel de submulțime DTD există. Deci orice alte declarații din submulțimea internă, inclusiv setarea textului de înlocuit al entităților generale interne, setarea valorilor prestabilite ale atributelor și normalizarea atributelor trebuie prelucrate și realizate. Figura 3.2 arată o reprezentare metaforică a diferenței dintre ceea ce vede un analizor XML care validează și unul care nu validează. Analizorul care validează vede clar totul. Analizorul care nu validează poate că este capabil, sau poate că nu, să vadă entitățile, dar în mod sigur nu vede DTD-ul, cu toate că știe că DTD-ul există.

Chiar dacă procesorul XML care nu validează trebuie să citească și să proceseze submulțimea internă a DTD-ului până când, și dacă, este necesar să oprească prelucrarea, el nu poate valida documentul pe această bază. Dacă o face, va fi un procesor care validează și va trebui să citească totul.

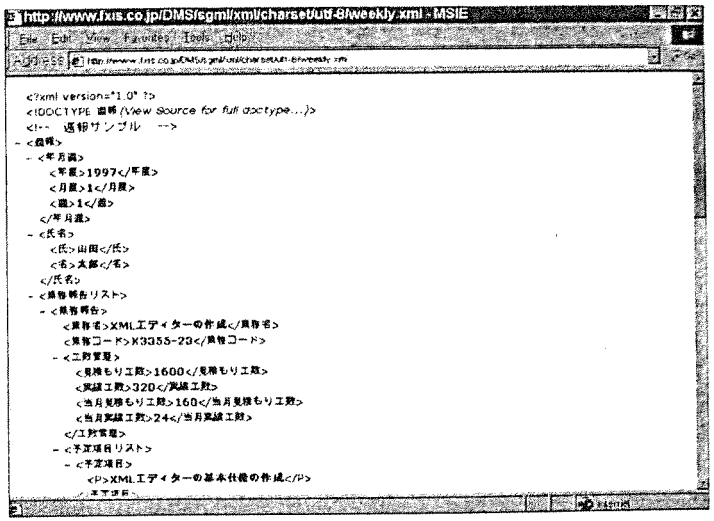


FIGURA 3.2
Este prezentată diferența dintre vizualizarea unui document XML de către un analizor care validează și un analizor care nu validează.

Notă Este surprinzător cât de mulți oameni înțeleg greșit entitățile parametru și aceasta este una dintre problemele cu EBNF care formează o parte a standardului XML 1.0. Programatorii sunt familiarizați cu EBNF și consideră că trebuie să definească totul, dar o mare parte a specificației este deja conținută în textul deseori neclar care o însoțește. Trebuie să citiți atât EBNF-ul cât și textul pentru a înțelege pe deplin sensul Recomandării W3C.

Entități generale

O *entitate generală* poate apărea în documentul însuși, cel puțin în mod potențial. Ea este identificată printr-o sintaxă particulară în declarație:

```
<!ENTITY nume {informatii} >
```

Marea diferență între entitățile generale este că sunt fie interne, caz în care `informatii` este un șir încadrat de ghilimele, fie externe, reprezentând fie o intrare din catalog, fie un URL.

```
<!-- Entitate generala interna -->
<!ENTITY name "un fel de text" >
<!-- Entitati generale interne -->
<!ENTITY nume PUBLIC "-//LeeAnne.com//My XML Stuff//EN" >
<!ENTITY nume PUBLIC "-//LeeAnne.com//My XML Stuff//EN" "my-dtd=stuff.dtd" >
<!ENTITY nume SYSTEM "http://www.leeanne.com/xml/my-xml-stuff.dtd" >
```

Entitățile externe pot să nu fie incluse în document dacă procesorul XML nu validează. Entitățile generale interne *pot* apărea în DTD, dar numai ca valori într-o listă de atribute sau într-o entitate. În mod fundamental, aceasta înseamnă că ele trebuie să apară ca șiruri încadrate între ghilimele.

Entități neanalizate

Entitatea neanalizată a fost deja tratată anterior, la explicațiile despre notații. Entitățile neanalizate pot fi numai externe, dar externalitatea entității cade în grija declarației

notației referită ei într-un atribut NDATA. Utilizarea lor, ca și cea a notațiilor în general, este întrucâtva controversată. Nu există nici un motiv pentru care proiectantul unui DTD ar trebui să știe ce tip anume de fișier multimedia va fi în interiorul unui document și apoi să dirijeze pe nevăzute agentul corespunzător. În loc să fie un șablon de document general, atunci DTD-ul va fi limitat la tipurile fișierelor neanalizate prevăzute de la început și de care se va ține cont.

Acest lucru este în contrast cu cazurile existente în HTML. În cadrul limitelor, indicați spre un fișier binar, iar aplicația își dă seama de ce tip este și cum să îl afișeze. Este altfel decât în cazul mediului UNIX, de unde provin marea majoritatea a proiectanților XML. În cadrul limitelor, în UNIX utilizați doar un fișier. Fișierele executabile se autoidentifică și se comportă corect pe cont propriu. După umila mea părere, aceasta pare o abordare mult mai robustă.

Indiferent de situație, rămâneți cu dificila sarcină de a actualiza DTD-urile dumneavoastră ori de câte ori este inventat un nou format grafic sau audio. Alternativa este să lăsați DTD-ul în pace și să rămâneți în urmă, pe măsură ce pozele sunt înlocuite de video, video este înlocuit de video interactiv și simpla interacțiune 2D este înlocuită de realitatea virtuală 3D.

Declarația ENTITY pentru o entitate neanalizată funcționează mână în mână cu entitatea notație. Notația (NOTATION) *trebuie* declarată înainte de utilizarea ei într-o declarație ENTITY. Aici vă este prezentat cum va arăta declarația unei entități neanalizate în DTD-ul dumneavoastră, împreună cu declarația elementului care este necesară pentru a instanția într-adevăr un anumit exemplu:

```
<!ENTITY gif89a SYSTEM "gif89a.gif" NDATA gif89a>
<!ELEMENT image EMPTY>
<!ATTLIST image source CDATA #REQUIRED
                alt    CDATA #IMPLIED
                type   NDATA gif89a >
```

În documentul dumneavoastră, eticheta va arăta astfel:

```
<image source=uri"
      alt=[o imagine]>
```

Puteți crea, de asemenea, o listă enumerată de tipuri de notații, care utilizează o sintaxă ușor diferită pentru a descrie tipul notației, după cum ați învățat în discuția despre declarațiile NOTATION:

```
<!NOTATION gif87a     PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
➥87a//EN" "gif">
<!NOTATION gif89a     PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
➥89a//EN" "gif">
<!ENTITY gif87a NDATA gif87a>
<!ENTITY gif89a NDATA gif89a>
<!ELEMENT image EMPTY>
<!ATTLIST image source CDATA #REQUIRED
                alt    CDATA #IMPLIED
                type   NOTATION (gif87a | gif89a) "gif89a" >
```

În documentul dumneavoastră, eticheta va arăta astfel:

```
<image source=uri"
      alt=[o imagine]>
```


dar puteți să nu luați în seamă tipul prestabilit dat în lista de atribute, dacă gif-ul dumneavoastră era în formatul mai vechi gif87a:

```
<image source=uri"
      alt=[o imagine]
      type="gif87a" >
```

Acesta este un mic exemplu care evidențiază absurditatea acestei abordări. Câte persoane știu pe nepregătite cărui format din cele două îi aparține fișierul lor gif? Câtor persoane le pasă? XML în forma sa actuală face din aceasta și din multe alte lucruri lipsite de importanță o problemă de o importanță covârșitoare. În cuvintele nemuritoare ale lui Tim Bray, unul dintre membrii echipei de proiectare a XML 1.0, „Este în întregime fals".

Declarația ELEMENT

Declarația elementelor este acea parte a XML pe care o vedeți cel mai clar în produsul final. Ea reprezintă etichetele efective pe care le veți utiliza în documentele dumneavoastră, și trebuie să aveți cel puțin una pentru că, în caz contrar, documentul dumneavoastră nu este un XML valid. Un procesor XML care nu validează nu va vedea niciodată DTD-ul dumneavoastră, dar etichetele și atributele conținute în documentul dumneavoastră îl vor descrie, oricum, suficient de complet. Împreună cu o foaie de stil asociată, puteți afișa corect documentul, fără a avea nevoie de vreun DTD.

Dacă aspectul documentului nu vă interesează, poate că nici nu veți avea nevoie de o foaie de stil. Aceasta poate fi situația pentru un document care a fost în esență o bază de date sau a fost proiectat ca un mecanism de transport pentru transferul de date structurate între două aplicații.

Declarația ELEMENT arată astfel:

```
<!ELEMENT nume content-model >
```

nume este numele etichetei utilizate. content-model (modelul de conținut) este locul unde lucrurile încep să devină interesante.

content-model poate conține un amestec arbitrar de elemente terminale și neterminale. Elementele neterminale sunt numele altor elemente, în timp ce elementele terminale sunt texte sau alt conținut. Aceasta este sintaxa care formează nodurile în documentul dumneavoastră. Există două modele generale de conținut. Primul model descrie conținutul secvențial – sau ordonat – care utilizează o listă separată prin virgule pentru a indica faptul că un element trebuie să urmeze după altul până la sfârșitul listei ordonate. Cel de-al doilea model utilizează o bară verticală „sau" ca separator al listei pentru a indica o selecție dintre alternative. Cu aceste două mecanisme puteți construi aproape orice.

Conținut ordonat

Numele de entități separate prin virgule sunt ordonate secvențial. Primul din listă este primul, cel de al doilea din listă este al doilea și așa mai departe. Elementele listei trebuie, pentru claritate, încadrate între paranteze, în ciuda faptului că nu este strict necesar pentru o listă ordonată corect:

```
<!ELEMENT nume ( sub-element1, sub-element2, sub-element3, … ) >
```

Conținut de selecție

Numele entităților sunt separate prin bare „sau" (|), bare verticale care ar trebui să fie disponibile pe tastatura dumneavoastră specifică limbii naționale, deseori deasupra caracte-

rului backslash (\). Ele trebuie să fie încadrate între paranteze. În utilizare, ele arată astfel:

```
<!ELEMENT nume ( sub-element1 | sub-element2 | sub-element3 | … ) >
```

Repetarea elementelor de conţinut XML

Numele de conținut, sau grupurile de nume de conținut încadrate între paranteze, pot fi urmate de un semn de întrebare (?), un semn plus (+) sau un asterisc (*) pentru a arăta un factor de repetiție, numit uneori *indicator de apariții*. Lipsa unui semn de repetare indică faptul că elementul trebuie să apară o singură dată. Un semn de întrebare semnifică faptul că elementul poate să nu apară niciodată sau să apară o singură dată. Un semn plus înseamnă că elementul apare cel puțin o dată și, dacă este nevoie, se va repeta. Un asterisc înseamnă că elementul se repetă de câte ori este nevoie, dar este opțional. Cu alte cuvinte, un asterisc înseamnă zero sau mai multe apariții. Aceste semne pot fi combinate cu paranteze și cu secvențe „sau" alternative pentru a forma structuri de o complexitate arbitrară.

Modelele de conținut sunt atât de importante în XML încât poate că merită să le scrieți undeva până le veți memora. Tabelul 3.1 prezintă simbolurile utilizate pentru a arăta factorii de repetiție și cele două tipuri de modele de conținut.

Tabelul 3.1 Indicatorii de apariții utilizați în DTD-urile XML

Sintaxă	Semnificație
?	Zero sau o singură apariție
+	Una sau mai multe apariții
*	Zero sau mai multe apariții
(a | b)	Fie a, fie b, dar nu ambele
(a , b)	a urmat de b

Dacă sunteți obișnuit cu expresiile regulate din Vi, Emacs și alte editoare stil UNIX, sintaxa vă va fi destul de cunoscută. Exemplul următor prezintă diverse utilizări:

```
<!ELEMENT nume (( subelement1 | subelement2)? , (subelement3))+ >
```

Acesta spune că elementul conține una sau mai multe substructuri conținând fie `subelement1` fie `subelement2`, urmat de o instanță a lui `subelement3` sau conține o instanță a lui `subelement3`. Deci, următoarele producții sunt toate corecte:

```
<nume><subelement3></subelement3></nume>
<nume><subelement3></subelement3><subelement3></subelement3></nume>
<nume><subelement1></subelement1><subelement3></subelement3></nume>
<nume><subelement2></subelement2><subelement3></subelement3></nume>
```

Chiar și acest exemplu simplu poate genera un număr infinit de producții, în ciuda faptului că scrierea lor poate deveni plictisitoare. Modurile în care pot fi combinate aceste elemente simple pot deveni repede neclare. Una dintre utilizările cele mai prietenoase cu utilizatorul ale entităților parametru este încapsularea submulțimilor cu astfel de comportament, pentru ca dumneavoastră să vă puteți ocupa separat de ele.

Notă *Anticipare* este un termen din lumea compilatoarelor/analizoarelor care înseamnă pur și simplu că analizorul poate privi înainte în fluxul de intrare și să revină pentru a rezolva ambiguitățile. Deoarece aceasta implică abilitatea de a păstra în memorie, dacă este necesar, întregul document, orice depășește anticiparea cu un singur caracter – atât de obișnuită încât nici nu este demnă de numele de anticipare – necesară pentru a rezolva token-urile, a fost eliminat din definiția limbajului. Evitarea nivelurilor arbitrare de anticipare înseamnă că un analizor XML poate fi mic și de categorie ușoară, potrivit pentru un dispozitiv handheld și pentru alte dispozitive cu memorie și putere limitată.

Deoarece procesoarele XML nu anticipează, trebuie să garantați că modelul dumneavoastră de conținut poate fi analizat cu succes, fără reveniri, înainte de a-l transmite în continuare. O strategie bună este de a structura un model de conținut cu multe elemente opționale, ca o alternativă între modelele în cascadă cu elemente opționale eliminate de la începutul submulțimii modelului:

```
(a,b,c,d) | (b,c,d) | (c,d) | (d)
```

Nu puteți îndepărta elemente de la sfârșit sau să puneți alternative în mijloc, deoarece atunci analizorul ar trebui să revină pentru a le analiza. Deci modelele de conținut care seamănă cu următoarele exemple ambigue probabil că nu vor face ceea ce ar fi dorit proiectantul lor:

```
(a,b,c,d) | (a,b,c) | (a,b) | (a)
(a,b,c,d) | (a,b,d) | (a,d) | (a)
```

În ambele exemple, atunci când analizorul întâlnește elementul a, este aleasă prima alternativă, `(a,b,c,d)`. Celelalte alternative nu vor fi luate în considerare și pot fi ignorate. Unele analizoare XML pot genera o eroare atunci când întâlnesc un model de conținut nedeterminist, deci vi se cere să garantați că toate modelele de conținut sunt neambigue.

Notă Din punct de vedere tehnic, modelele de conținut ambigue prezentate aici sunt nedeterministe, ceea ce înseamnă că nu este posibil să construiți un automat cu stări finite care să le proceseze. Poate că este posibil să convertiți un model de conținut nedeterminist la unul care este determinist din punct de vedere algoritmic, dar acest lucru nu este garantat.

Prin imbricarea fragmentelor cunoscute de logică combinatorică în unele mai mari, ceea ce poate fi intimidant atunci când este vizualizat în întregime poate fi desfăcut în părți componente. Din nefericire, din cauza limitărilor submulțimii interne a DTD-ului, această facilitate este disponibilă numai în DTD-urile externe.

Conţinut terminal

Frunzele arborelui documentului dumneavoastră sunt reprezentate prin conținut terminal, din care avem un singur tip: `#PCDATA`. *Datele caracter analizate* (*Parsed Character Data*) sunt formate din conținut combinat care poate conține atât text, cât și marcaje. Este cel mai general tip de frunză. Atunci când utilizați un model de conținut combinat nu puteți controla ordinea sau aparițiile elementelor, cu toate că puteți constrânge elementele să fie de un anumit tip.

Într-un element va fi utilizat ca

```
<!ELEMENT nume (#PCDATA | el1 | el2 | el3 | … )* >
```

sau ca mai jos, dar fără a avea control asupra tipului elementului:

```
<!ELEMENT nume (#PCDATA)* >
```

Este destul de simplu.

Sugestie de la LeeAnne.com Words to weave by

Puteţi utiliza acest tip de conţinut de element pentru a include etichete XML dintr-o altă zonă de nume, de exemplu, într-un document XML, fără a viola DTD-ul documentului de bază. Cu toate că DTD-ul documentului nu va avea nici o idee despre ce înseamnă etichetele inserate, este de presupus că DTD-ul care administrează zona de nume şi proiectantul paginii vor şti.

Conţinut EMPTY

Dacă elementul este declarat ca fiind vid (EMPTY), atunci el nu are conținut. Deci dacă dumneavoastră utilizați convenția etichetă de pornire – etichetă de încheiere, atunci trebuie să garantați că eticheta de încheiere urmează imediat după eticheta de pornire, ca într-o declarație DTD: `<!ELEMENT oricenume EMPTY>`

instanță dintr-un document XML: `<oricenume></oricenume>`

sau, în caz contrar, veți genera o eroare de validare.

De asemenea, probabil că veți bloca orice browser care intră în el dacă se întâmplă ca numele să semene cu o etichetă vidă HTML. Deci acest lucru, în XML, este nesigur cu toate că este perfect legal:

```
<img></img>
```

La urma urmelor, probabil că este mai bine să utilizați o sintaxă specială pentru etichete vide, ca aceasta:

```
<oricenume />
```

Observați că există un spațiu între numele elementului și caracterul slash, iar acesta este urmat imediat de paranteza unghiulară închisă.

Notă

Deoarece elementele vide sunt prin definiţie vide, singurul conţinut posibil pe care îl pot avea sunt atributele asociate fiecărui element vid. De obicei, orice conţinut text poate fi inclus cu uşurinţă într-un atribut. Singura limitare reală este că nu puteţi extinde în mod tipic documentul în sine prin intermediul unui atribut. Există două excepţii importante: din punct de vedere teoretic, o notaţie poate apela un proces care să insereze mai mult conţinut, la fel cum se poate face utilizând Dynamic HTML, şi este posibil să se transcludă conţinut dintr-un fişier extern utilizând atribute la un element ancoră XLink. Pentru mai multe informaţii despre XLink, XPath şi XPointer, studiaţi Capitolul 8, „XPath", şi Capitolul 9, „XLink şi XPointer". Indirect, ar fi posibil să utilizaţi XSLT pentru a transforma un document bazat pe valoarea conţinută într-un atribut, dar aceasta va trebui să mai aştepte până când vom descoperi XSL în Capitolul 13, „Foi de stil în cascadă şi XML/XHTML".

Conţinut ANY

Dintre toate declarațiile studiate, aceasta este de neegalat în generalitate și înseamnă exact ceea ce spune. Un element definit astfel poate conține absolut orice:

```
<!ELEMENT oricenume ANY>
```

Este echivalentul trecerii fiecărui element în DTD-ul dumneavoastră, în orice ordine, dar salvează mult timp necesar pentru introducerea datelor.

Conținutul ANY este comod în primul rând pentru dezvoltarea unui DTD sau pentru depanarea unui DTD stricat, dacă aveți un document exemplu, dar nu și un DTD. Dacă aveți erori de validare, o primă încercare ar fi să modificați modelul de conținut la ANY în locurile probabile până când DTD-ul este valid și îl puteți încărca într-un instrument de proiectare DTD. În acest punct puteți începe restrângerea modelului dumneavoastră de conținut. După aceasta veți avea un DTD valid. Cu un model de document mare, acest lucru poate fi obositor, dar este echivalentul tricotatului pentru un proiectant DTD; după un timp, procesul devine atât de mecanic încât nici nu vă mai gândiți la el.

Crearea declaraţiilor ATTLIST

ATTLIST este un container pentru atributele asociate unui element. Are o sintaxă simplă, dar alternativele pot fi complexe și greu de urmărit:

```
<!ATTLIST nume-element nume-atribut-1 tip-data data-prestabilita
                       nume-atribut-2 tip-data data-prestabilita
                       … >
```

Următoarele subsecțiuni definesc ce tipuri de date pot fi utilizate într-o listă de atribute. Pentru analizoarele care nu validează, aceste tipuri de date sunt oarecum irelevante deoarece analizorul nu va ști nimic despre ele. Dacă interesul dumneavoastră stă în prelucrarea ulterioară a documentelor XML utilizând analizoare care nu validează, atunci puteți sări peste această întreagă secțiune fără a avea multe de pierdut.

Tipuri de date XML

În atribute sunt permise mult mai multe tipuri de date decât în modelele de conținut. În timp ce un model de conținut permite numai PCDATA, atributele pot lua tipuri de date destul de specifice. Tipul de date al atributelor XML definește în termeni generali ce fel de informație poate conține o entitate și cum va fi structurată aceasta. Tipul de date al atributelor este luat din lista următoare:

- CDATA – Data de tip caracter este un șir.
- ID – Un ID unic, simbolic, dintr-un document. Cu toate că vi se permite să declarați o valoare prestabilită, de fapt nu aveți de ce să faceți acest lucru. De asemenea, deoarece aveți nevoie de acces la DTD pentru a determina dacă există sau nu un ID conținut în element, acesta este inutil pentru procesoarele care nu validează. Majoritatea instrumentelor care depind de ID-uri plasează un ID teoretic cu numele "id" care există în interiorul fiecărui element și îl utilizează fie că este declarat, fie că nu.
- IDREF – O referință simbolică la un ID unic. Vezi ID.
- IDREFS – O listă de referințe simbolice la ID-uri unice dintr-un document. Vezi ID.

- ENTITY – O referință simbolică la o entitate care referă o notație. Acesta este un alt mod de a referi date neanalizate.
- ENTITIES – O listă de referințe simbolice la entități care referă notații. Acesta este un alt mod de a referi date neanalizate.
- NMTOKEN – O referință simbolică la un simbol NAME.
- NMTOKENS – O referință simbolică la o listă de simboluri NAME.
- NOTATION – O referință enumerată la o listă de tipuri de date notație. Aceasta este o excepție de la regula că toate entitățile neanalizate trebuie să fie externe. Dacă pentru un element definiți un atribut de tipul NOTATION, puteți de fapt să puneți notația dumneavoastră in-line în documentul dumneavoastră. Presupunând că veți defini tipul atributului ca NOTATION în următorul exemplu XML, veți fi capabil să includeți JavaScript fără o referință externă, dacă și numai dacă utilizați un analizor care validează.

```
<Script type="JavaScript"> … Cod JavaScript … </Script>
```

A spune că va trebui să fiți atent atunci când faceți acest lucru este numai pe jumătate adevărat. Observați că nu este o eroare dacă un agent utilizator extern eșuează în încercarea de a înțelege o notație, deci JavaScript-ul poate fi redat ca un simplu text sau poate fi ignorat complet, în funcție de combinația analizor și agent utilizator.

- (attr1 | attr2 | …) – O listă enumerată de valori posibile pentru atribute. Multe medii de creare XML le afișează ca liste de selecție autoderulante sau pop-up pentru a fi utilizate mai ușor.

Deoarece într-un singur element pot exista mai multe atribute, un ATTLIST poate deveni complex. Oricum, toate se supun aceluiași set de reguli, deci printr-o studiere atentă ar trebui să vă reveniți după orice confuzie inițială.

Subsecțiunile următoare studiază mai detaliat conținutul atributelor, începând cu sintaxa valorii prestabilite.

Specificarea datelor prestabilite dintr-un atribut

Poziția de completat data-prestabilita din șablonul listei de atribute prezentat anterior în această secțiune poate fi #REQUIRED, #IMPLIED, o valoare prestabilită sau #FIXED și o valoare constantă.

Acestea sunt exemple care funcționează:

```
<!ATTLIST numeelement  attr1 #REQUIRED
                       attr2 #IMPLIED
                       attr3 "valoare-prestabilita"
                       attr4 #FIXED "valoare constanta">
```

Atât #REQUIRED cât și #IMPLIED nu au valori prestabilite deoarece ar fi fără sens în acest context. Dacă atributul este #REQUIRED, el trebuie să fie acolo fie că avem sau nu o valoare prestabilită. Dacă atributul este #IMPLIED, atunci atributul nu este necesar și nu trebuie să fie prezent.

Notă Dacă a existat vreodată un termen confuz, #IMPLIED este acela. El sună aproape ca și „prestabilit". Poate că ați crezut în mod naiv că valoarea „implicită" (implied) este cea prestabilită. Dar nu este. *Implied* este acolo pentru a spune procesorului că nu trebuie să își bată capul căutând un atribut, deoarece este posibil să nu fie acolo. Alegerea cuvintelor a fost o nefericită deviere de la claritate și nimic mai mult.

IMPLIED este o relicvă din zilele SGML-ului, când descrierile simple erau uneori sacrificate pentru un termen „logic", precis, a cărui origine zace ascunsă în mintea prăfuită a unui programator din primele zile ale informaticii. După părerea mea, cuvântul este un simplu paronim deoarece un termen oarecum înrudit și deseori confundat, *inferred* (*dedus*), descrie precis ceea ce ar trebui să facă procesorul cu un atribut „implicit", adică să deducă existența sau inexistența unui atribut bazându-se pe prezentarea elementului în documentul XML însuși.

Atribute de tipul CDATA

Acesta este *foarte* simplu. Este un șir de caractere. Exemplul următor definește un atribut numit sir, a cărui valoare este CDATA, și o valoare prestabilită pentru acel șir:

```
<!ATTLIST nume sir CDATA "informatii sir">
```

Observați că, așa ca de fiecare dată, analizoarele care nu validează nu vor citi definiția sau valoarea prestabilită a DTD-ului extern.

Atribute de tipul ID

Acestea se referă la ID-uri unice din documentul dumneavoastră și furnizează o metodă ușoară de a vă asigura că există un nume disponibil pentru a forma celălalt capăt al unei legături. Cu toate că este un lucru inteligent să faceți așa, în practică, majoritatea browserelor probabil că nu vor valida și nu vor vedea niciodată definiția ID-ului. Majoritatea instrumentelor pointer actuale cer prezența unui atribut de tipul ID numit id sau name indiferent dacă este sau nu în DTD, lăsându-le astfel să genereze liber legături.

Codul următor definește un atribut ID numit nod și apoi prezintă cum poate fi utilizat atributul într-un document XML.

declarație DTD: `<!ATTLIST name nod ID #REQUIRED>`

instanță dintr-un document XML: `<name nod="A1234">`

O cale mai sigură ar fi să definiți atributul ID cu numele id, așa încât analizoarele care nu validează să poată să îi deducă intenția, așa cum fac browserele HTML cu atributul name. În secțiunea următoare veți vedea două moduri de a face acest lucru.

Atribute de tipul IDREF

Aceasta este o metodă de a vă referi la celălalt capăt al unei perechi id/idref. IDREF indică spre id.

declarație DTD: `<!ATTLIST oricenume pointer IDREF #REQUIRED>`

instanță dintr-un document XML: `<oricenume pointer="A1234">`

În termeni HTML, acesta este echivalentul brut al atributului href care indică spre o ancoră cu nume.

Notă ID-urile sunt atât de comode încât este o rușine că trebuie să le treceți peste tot, în special atunci când nu aveți acces la scriere în documentul sursă. Generația următoare de mecanisme de legare, XLink și XPointer XML, vă vor permite să utilizați automat structura documentului pentru a obține aproape același rezultat și în documentul afișat.

Într-o lume ideală, ID-urile sunt articolele care aproape sigur că vor rămâne staționare, măcar aproximativ, într-un document. Din acest motiv ele sunt o țintă bună pentru pointeri. Fragmentul de sintaxă URL#namevalue utilizat în HTML pentru a indica locații din document funcționează și în XML, singura excepție fiind aceea că ținta trebuie să fie un ID.

Corolarul la aceasta este că ambele atribute, ID și name, trebuie declarate pentru fiecare țintă posibilă, deoarece agenții utilizator moșteniți vor găsi elementul conținând atributul name indicat, iar agenții utilizator cu suport XML îl vor găsi pe ID. Puteți, din punct de vedere teoretic, să comasați cele două într-un singur atribut prin declararea unui atribut name ca fiind de tipul ID, dacă știți că documentul va fi vizualizat utilizând numai analizoare XML care validează. Din păcate, eticheta ID este de asemenea utilizată în HTML, într-un fel de schizofrenie rezultată din încercarea de a păstra sintaxa veche și de a crea una nouă, care să fie mai sugestivă. Deci, este mai bine să declarați două atribute cu aceeași valoare. Atributele de tipul ID trebuie să fie unice în cadrul unui document, iar atributele name ar trebui să dubleze atributul de tip ID.

Deci, în mod ideal, ar trebui definite atributele următoare în orice element care ar putea fi utilizat ca țintă a unei hiperlegături:

```
<!ATTLIST oricelegatura id ID          #IMPLIED
                           name NMTOKEN #IMPLIED
                           alte-atribute … >
```

Dacă doriți să forțați prezența celor două atribute legătură la un element, le puteți defini după cum este necesar, astfel:

```
<!ATTLIST oricelegatura id ID          #REQUIRED
                           name NMTOKEN #REQUIRED
                           alte-atribute … >
```

Vă rog să observați că atributul name nu poate fi de tipul ID, cu toate că ar trebui să dubleze conținutul atributului id, din moment ce analizorul va impune unicitatea atributului ID, împiedicându-vă să declarați o a doua instanță a unui atribut de tipul ID. În utilizare, puteți instanția ținta unei hiperlegături astfel:

```
<oricelegatura id="A1234" name="A1234">continutoricetinta</oricelegatura>
```

Dacă s-a ales ca numele elementului *oricelegatura* să fie a, atât browserele exclusiv HTML cât și cele cu suport XML vor găsi aceeași locație.

```
<a id="A1234" name="A1234">continutoricetinta</a>
```

Aceasta este strategia utilizată de XHTML, deoarece este proiectat, în parte, atât să fie accesibil pentru cât mai multe browsere HTML moștenite, cât și să furnizeze avantajele extensibilității pentru browserele cu suport XML.

Atribute de tipul ENTITY

Iată cum veți utiliza un atribut de tipul ENTITY pentru a referi o notație:

```
<!NOTATION gif PUBLIC "…">
<!ENTITY gifpicture SYSTEM "pozamea.gif" NDATA gif>
<!ELEMENT picture EMPTY>
<!ATTLIST picture src ENTITY #REQUIRED>
```

Într-un document XML bun veți utiliza eticheta picture definită în DTD astfel:


```
<picture src='gifpicture'>
```

Aceasta este încă o metodă prin care XML cere ca dumneavoastră să fiți un guru multimedia. Cu toate că sunt sigură că există diferențe importante între diversele metode disponibile pentru a face acest lucru, întrebarea reală este dacă trebuie de fapt făcut. Vedeți comentariile mele referitoare la notații în general, prezentate anterior în acest capitol.

Notă

Nu-mi place să insist asupra acestui lucru și promit să nu o mai fac după acest capitol, dar a sili procesorul XML să știe despre toate tipurile multimedia posibile nu este o metodă robustă de realizare a lucrurilor. În primul rând, înseamnă că nu puteți insera o imagine sau un fișier multimedia într-un document decât dacă puteți garanta că toți cei care îl vor utiliza vor folosi analizoare XML care validează. Această condiție pare excesiv de restrictivă. Grupul de lucru XML al W3C pare să fi ignorat exemplele de plug-in-uri pentru browsere și redirecționările server-side, care ar fi făcut posibil să negociați cu agentul utilizator pentru a stabili aptitudinile sale și să furnizați, eventual, un fișier multimedia corespunzător. Dacă nu, agentul utilizator va ști că trebuie să găsească de unul singur un asistent corespunzător, așa cum se întâmplă acum cu tipurile de conținut MIME nerecunoscute. Trebuia furnizat un mecanism mai bun. Negocierea conținutului între browser și server ar fi metoda modernă de furnizare a acestei flexibilități, deoarece mediile diferite ale utilizatorilor pot necesita formate de fișiere total diferite.

Acestea fiind zise, convenția *funcționează*, chiar dacă este incomodă, și poate va fi înlocuită în viitor cu un mecanism simplu care ignoră mașinațiunile existente în favoarea unei alte metode de identificare a tipului de conținut. Standardele W3C pentru URI-urile persistente descurajează cu putere legarea locațiilor de tipul de date și de metodele de acces, care sunt violate de sintaxa existentă a notațiilor.

Deoarece este posibil ca tipurile de date să se schimbe în viitor, W3C recomandă ca negocierea conținutului să aibă loc între server și agentul utilizator, pentru a determina ce tipuri de date sunt suportate și ce tipuri sunt disponibile. În acest caz, puteți referi o notație ca un nume simplu, cum ar fi „pozamea", și să lăsați serverul să își dea seama ce gen de imagine este disponibilă pentru acel agent particular și să o furnizeze, spunând agentului ce a furnizat ca să nu apară greșeli. După cum se spune, totul se va rezolva până la urmă, deci puteți sta liniștit deoarece în mod sigur situația se va schimba cândva în viitor.

Atribute de tipul NMTOKEN

Conținutul unui atribut este constrâns să arate ca un nume simbolic, după cum este definit în Recomandarea XML. Cu alte cuvinte, el este format dintr-o literă Unicode/ISO, o ideogramă, o liniuță de subliniere sau două puncte urmate de litere, ideograme, cifre, puncte, cratime, liniuțe de subliniere sau două puncte, repetate de câte ori doriți, în orice combinație. Mai simplu, aceasta înseamnă că un nume nu poate începe cu o cifră, un punct sau o cratimă, cu toate că le poate conține pe oricare dintre acestea, plus liniuțe de subliniere sau două puncte, după caracterul inițial, dar nu și alte simboluri tipografice cum ar fi asteriscuri, slash-uri, linii, ampersand-uri sau altele de acest gen. Expresia „litere și ideograme" conține caracterele pe care cineva le-ar utiliza de obicei pentru a scrie cuvinte într-o anumită limbă. De exemplu, în chineză acestea ar putea fi caractere simple indicând cuvinte complete; în japoneză, caractere indicând silabe complete; și în multe limbi, litere individuale indicând consoane și vocale care sunt combi-

nate pentru a alcătui cuvinte. Combinațiile posibile ale acestora extind înțelesul numelor pentru a include aproape întreaga populație alfabetizată a lumii, în orice limbă omenească.

Atenție După cum s-a menționat anterior, utilizarea a două puncte este puternic combătută deoarece pot fi încurcate ușor cu prefixul unei zone de nume. De asemenea, șirul "xml", în orice combinație de tipuri de litere, este rezervat.

Acestea sunt NMTOKEN-uri:

```
nume
nume_nou
_nume.sufix-nume
nume.sufix
:nume
baloane99
nume1.parte_sufix
```

iar acestea nu:

```
nume&altele
99baloane
.nume
$valoare
nume(functie)
```

Deși puține texte în XML, cel puțin în engleză, demonstrează utilizarea ideogramelor și a script-urilor non-latine într-un nume de element, printre alte motive din cauza dificultății de scriere a lor, posibilitatea există și va fi utilizată considerabil în acele domenii unde este nevoie de ea. Figura 3.3 prezintă un exemplu de document XML în limba japoneză, disponibil pe Web, în situl FujiXerox la `http://www.fxis.co.jp/DMS/sgml/xml/charset/utf-8/weekly.xml`.

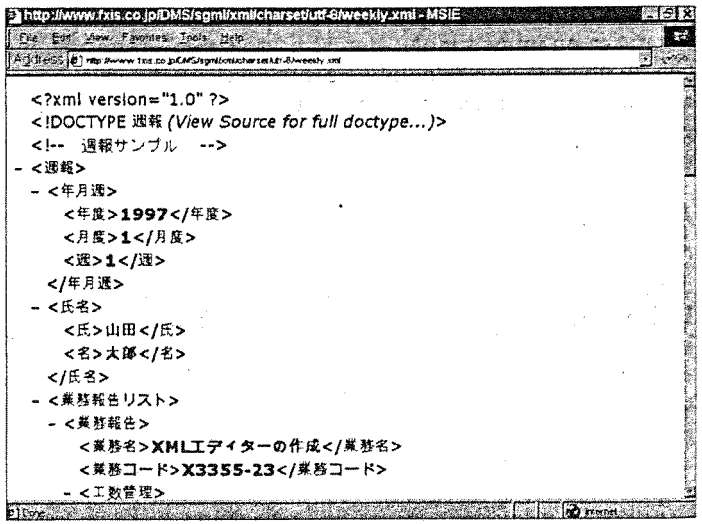


FIGURA 3.3
Un exemplu de document în limba japoneză, marcat cu etichete XML japoneze sugestive.

Legătura la acest exemplu a fost furnizată cu amabilitate de către situl *Chinese XML Now!* al Academiei Sinica, la `http://www.ascc.net/xml/`. Situl conține numeroase legături

la informații de interes pentru dezvoltatorii de documente XML în chineză, seturi de testare pentru a confirma aptitudinea de a manevra corect intrările în limba chineză și exemple de documente XML în limba chineză.

Atribute de tipul NOTATION

Acestea trebuie să se potrivească cu definiția NMTOKEN și sunt date într-o listă simplă de alternative care definește enumerarea tipurilor posibile. Dintre acestea poate fi ales unul și numai unul. O valoare prestabilită poate fi specificată, dar nu este obligatorie. În utilizare, acestea arată astfel:

```
<!ATTLIST nume tip (gif87a|gif89a) >
```

Aceeași enumerare NOTATION va arăta astfel, atunci când va fi definită cu o valoare prestabilită:

```
<!ATTLIST nume tip (gif87a|gif89a) "gif89a" >
```

Toate notațiile trebuie declarate înainte de a fi utilizate într-o listă de atribute.

Atribute enumerate

Atributele enumerate arată ca și cele din exemplele anterioare și sunt definite utilizând o listă de alternative, dar nu referă notații. Ele sunt comode pentru crearea listelor de opțiuni. Majoritatea mediilor de editare XML vă oferă o mică listă autoderulantă pentru alegerea valorii dorite. Codul următor prezintă un atribut enumerat simplu, așa cum este definit într-un DTD:

```
<!ATTLIST name choice (yes|no) >
```

Asemenea atributelor NOTATION, atributele enumerate pot avea o valoare prestabilită:

```
<!ATTLIST name choice (yes|no) "yes">
```

Fragmentul de DTD de mai sus definește un atribut, choice, care poate lua valorile "yes" sau "no" și a cărui valoare prestabilită este "yes".

Utilizarea construcțiilor de tipuri de date XML speciale

Deși construcțiile din subsecțiunile următoare apar mai degrabă în document decât în DTD, funcțiile pe care le deservesc sunt destul de înrudite cu conceptele discutate deja în acest capitol.

Instrucțiunile de prelucrare permit autorului documentului să includă în document text care nu ar trebui analizat sau afișat de către analizorul XML sau de către agentul utilizator. În schimb, ele ar trebui transferate unei aplicații speciale de asistență, definită chiar în instrucțiunea de prelucrare. Ele sunt folosite cam pentru ce este folosit un element <script> în HTML, dar sunt mult mai generale. Observați în special faptul că ele nu folosesc etichete comentariu SGML pentru a ascunde text. Acest lucru nu este recomandat în XML deoarece analizoarelor XML nu li se cere să transfere textul comentariilor agentului utilizator.

Secțiunile CDATA constituie o metodă rapidă de a trece într-o secvență escape un întreg bloc de text, deci orice caractere de marcare incluse sunt tratate ca text, și nu ca marcaj.

Asemenea unei descrieri de conținut CDATA într-un DTD, ele spun analizorului că nu trebuie să încerce să analizeze textul conținut în element.

Instrucțiuni de prelucrare

O instrucțiune de prelucrare este transferată direct aplicației, fără a se încerca evaluarea ei. Ea începe cu <? și se încheie cu ?> și nu poate conține caracterele xml în nici o combinație de tipuri de litere.

Sugestie de la LeeAnne.com Words to weave by

Multe persoane au tânjit după un mecanism care să includă date binare direct în fișierul lor XML. Nu mai trebuie așteptată încărcarea referințelor externe și nu mai trebuie să vă întrebați dacă datele mai sunt acolo unde le-ați lăsat ultima dată. Întregul fișier este un glob mare de date. În teorie, aceasta este o idee bună, dar garantarea faptului că șirurile de încheiere nu apar în format binar este o problemă dificilă și probabil am cere prea mult ca un procesor XML să o rezolve.

Utilizând o instrucțiune de prelucrare, puteți transfera date direct unui anumit interpretor:

```
<?javascript {date javascript} ?>
<?Tcl {date Tcl} ?>
...
```

Avantajul este că puteți transfera liber date unui interpretor sau program în timpul rulării, fără a mai parcurge procesul de adăugare a unei notații în DTD-ul dumneavoastră. Evident, datele procesorului dumneavoastră nu trebuie să conțină șirul ?> fără să îl treacă într-o secvență escape.

Ascunderea într-o secvență escape a unui bloc de text într-o secțiune CDATA

O secțiune CDATA este un avantaj care vă permite să plasați o porțiune de date de tip caracter în documentul dumneavoastră fără truda de a include în secvențe escape fiecare caracter de marcare – sau ceea ce ar părea la prima vedere caractere de marcare – pe care îl conține.

În utilizare, aceasta arată astfel:

```
<![CDATA[Orice fel de text vechi inclusiv semnele < si & si chiar exemple de
➥etichete XML <!ENTITY lista … >]]>
```

O etichetă CDATA este un avantaj. Puteați face același lucru prin includerea în secvențe escape a tuturor caracterelor speciale, astfel

```
Orice fel de text vechi inclusiv semnele &lt; si &amp; si chiar exemple de
➥etichete XML &lt;!ENTITY lista …
```

dar ar fi durat probabil mult mai mult și exista o mult mai mare probabilitate de a greși. Singura constrângere este aceea că datele dumneavoastră nu pot conține secvența]]> care să încheie secțiunea. Figura 3.4 prezintă utilizarea secțiunii CDATA.

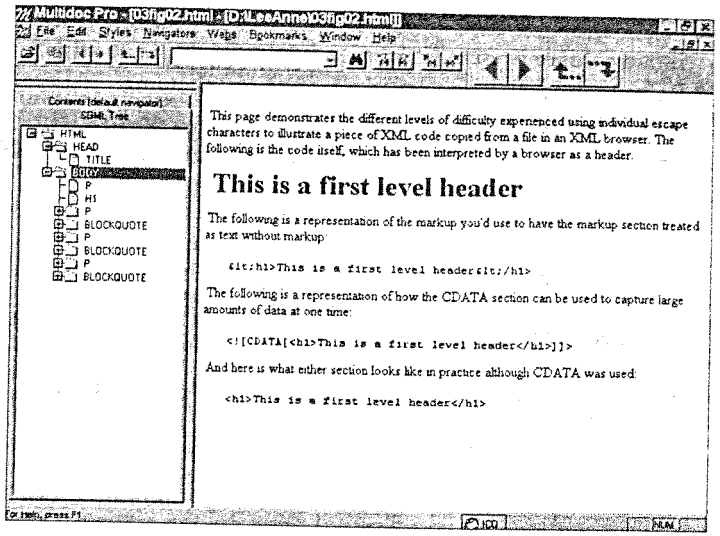


FIGURA 3.4
O pagină XML prezentând utilizarea unei secțiuni CDATA, pentru a evita secvențele escape dintr-o secțiune de marcare, în browserul SGML/XML MultiDoc Pro.

Diferența dintre un XML construit corect și un XML valid

Diferența dintre un XML construit corect și un XML valid este simplă: pe lângă faptul că este construit corect, un XML valid are asociat un DTD și a fost verificat în conformitate cu regulile conținute în DTD. Pe de altă parte, un XML doar construit corect nu este în mod *necesar* valid, deși ar putea fi.

Un XML construit corect respectă următoarele reguli:

- Etichetele trebuie să se imbrice corect. Fiecare pereche de etichete de pornire și de încheiere trebuie să conțină complet orice pereche de etichete care începe în interiorul ei. Cu alte cuvinte, nici o etichetă de pornire, etichetă de încheiere, etichetă element vid, comentariu, instrucțiune de prelucrare, referință caracter sau referință entitate nu poate începe într-o entitate și să se încheie în alta.
- În submulțimea internă a DTD-ului, referințele entitate parametru pot apărea numai la nivelul superior, unde pot apărea declarațiile de marcaj, și nu în declarațiile de marcaj. Entitățile parametru nu au restricții în submulțimea externă a DTD-ului.
- Numele din eticheta de încheiere a unui element trebuie să corespundă cu tipul de element din eticheta de pornire.
- Orice nume de atribut poate apărea o singură dată în aceeași etichetă de pornire sau element vid.
- Valorile atributelor nu pot conține referințe entitate directe sau indirecte la entități externe.
- Textul de înlocuit al oricărei entități referite direct sau indirect în valoarea unui atribut (altul decât <) nu poate să conțină semnul <.
- Caracterele referite utilizând referințe caracter trebuie să fie caractere permise.
- Declarația unei entități parametru trebuie să preceadă orice referință la ea.

- Declarația unei entități generale trebuie să preceadă orice referință la ea care apare într-o valoare prestabilită a unei declarații de listă de atribute.
- Referințele entități parametru pot să apară numai în DTD și au restricții în submulțimea internă a DTD-ului.

Singura metodă concretă de a spune dacă aveți un document XML valid este să utilizați un instrument automatizat pentru a citi documentul, inclusiv DTD-ul său, și să permiteți instrumentului să îl analizeze. Verificarea manuală poate ajuta, dar este notoriu faptul că e predispusă la erori, chiar dacă sunteți atent. De obicei este nevoie de două, trei sau chiar mai multe încercări pentru ca un DTD și un tip de document nou să se încarce la prima utilizare. Uneori sunt necesare mult mai multe încercări, deci aveți răbdare și perseverați, fiindcă *veți reuși*.

Modul de utilizare a DTD-urilor externe și a fragmentelor DTD

Unul dintre punctele forte ale XML este că puteți utiliza sau reutiliza tipurile documentului definite pentru un document la oricâte documente doriți. DTD-urile pot fi stocate într-un depozit central și pot fi chiar combinate pentru a realiza DTD-uri mari prin alegerea de secțiuni modulare.
Subsecțiunile următoare descriu cum pot fi accesate DTD-urile din documente XML sau din DTD-urile care le-au definit.

Indicarea unui DTD extern

DTD-urile externe care nu sunt locale pot fi indicate utilizând o declarație `DOCTYPE` ca aceasta, dacă DTD-ul este pe Web:

```
<!DOCTYPE article PUBLIC "-//LeeAnne.com//Article DTD//EN"
"http://www.leeanne.com/XML/article.dtd">
```

Sau, o serie de fragmente de DTD-uri poate fi indicată utilizând referințe entități parametru ca mai jos:

```
<!DOCTYPE article PUBLIC "-//LeeAnne.com//Article DTD//EN"
"http://www.leeanne.com/XML/article.dtd"> [
<!ENTITY % header PUBLIC "-//LeeAnne.com//Header DTD//EN"
"http://www.leeanne.com/XML/header.dtd">>
%header;
<!ENTITY % footer PUBLIC "-//LeeAnne.com//Footer DTD//EN"
"http://www.leeanne.com/XML/footer.dtd">>
%footer;
…
]>
```

Acest mecanism este foarte mult utilizat pentru a apela fișiere de referințe entitate caracter, dar poate fi utilizat și pentru alte scopuri. Fiți atent, totuși, deoarece procesoarelor care nu validează le este interzis să interpreteze orice entitate analizată care urmează după o referință la o entitate parametru externă care nu este citită.
Motivul este că starea fiecărei entități analizate este nedefinită după orice referință externă care nu este încorporată în document. De fapt, este corect ca un procesor care nu validează să ignore total entitățile externe.

Indicarea unui DTD local

Cu toate că un DTD local este, de asemenea, și un DTD extern, pentru referirea unui DTD local se utilizează o sintaxă ușor diferită deoarece nu trebuie să includă de regulă o referință într-un catalog. DTD-urile locale pot fi indicate utilizând declarația `DOCTYPE` astfel, dacă DTD-ul se află pe hard-discul local:

```
<!DOCTYPE article SYSTEM "article.dtd">
```

O serie de fragmente de DTD-uri poate fi indicată utilizând referințe entități parametru ca mai jos:

```
<!DOCTYPE article SYSTEM "article.dtd"> [
<!ENTITY % header SYSTEM "header.dtd">>
%header;
<!ENTITY % footer SYSTEM "footer.dtd">>
%footer;
…
]>
```

Acest mecanism este foarte mult utilizat pentru a apela fișiere de referințe entitate caracter, dar poate fi utilizat și pentru alte scopuri. Fiți atent, totuși, deoarece procesoarelor care nu validează le este interzis să interpreteze orice entitate analizată care urmează după o referință la o entitate parametru externă, inclusiv un DTD, care nu este citită.
Motivul este că starea fiecărei entități analizate este nedefinită după orice referință externă care nu este încorporată în document, deoarece o valoare poate să fie sau nu setată pentru aceasta în entitatea externă care nu este citită. Un analizor care nu validează nu are de unde să știe aceste lucruri. De fapt, este corect ca un procesor care nu validează să ignore total entitățile externe.

Sugestie de la LeeAnne.com Words to weave by

De ce ne batem capul, totuși, cu procesoarele care nu validează? Nu ar fi mai bine să validăm *totul*? Într-un singur cuvânt, nu! Închipuiți-vă cât de dificil ar fi să încărcați o pagină dacă browserele ar trebui să traverseze și să verifice fiecare legătură înainte de a evidenția o hiperlegătură.
Browserele XML pot *transclude*, include ca și conținut efectiv, documente de oriunde de pe Web. Deci nu aveți control asupra modului de structurare a paginilor. Dacă o pagină pe care o transcludeți include acces extensiv și recursiv la alte pagini sau DTD-uri, este posibil să aveți mult de așteptat până să fie încărcată pagina. Este probabil că browserele XML marchează locul unde este făcută o referință externă și permit utilizatorului să aleagă dacă să o încarce sau nu, la fel cum utilizatorul poate sau nu să urmeze o hiperlegătură.

Probabil că majoritatea agenților utilizator actuali (browsere) utilizați pe Web nu validează. Costurile potențiale ale validării, pentru vizitarea fiecărei locații referite, sunt atât de mari încât cel mai rațional plan pentru oricare browser este să amâne vizitarea documentelor externe până când utilizatorul cere acest lucru.

Utilizarea fragmentelor de DTD-uri

Ambele exemple precedente utilizau fragmente de DTD-uri pentru a extinde DTD-ul `article`. Dacă vă gândiți la un document ca la un arbore, atunci un fragment de DTD

este o metodă de a adăuga o altă ramură la arbore. DTD-ul trebuie structurat într-un mod în care acest lucru să poată fi făcut acordând o mare atenție zonelor de nume, până când inițiativa zonelor de nume XML va da rezultate.

Cea mai obișnuită utilizare a fragmentelor de DTD-uri este de a referi listele lungi de entități generale utilizate pentru a referi entități caracter prin nume mnemonice. Deci în loc de ‘ puteți folosi mult mai inteligibilul ‘ atunci când vă referiți la un apostrof stâng. Puțini oameni au învățat pe dinafară, după număr, tabelul de caractere ISO și Unicode. Este prea greu pentru majoritatea oamenilor. Cea mai simplă metodă de a utiliza mnemonicile este să apelați după nume un set predefinit al lor. Acestea sunt în număr foarte mare, evident, dar veți vedea că cele mai obișnuite sunt Latin-1, Special Symbols (simboluri speciale), Mathematics and Greek Symbols (simboluri matematice și grecești).

Iată cum puteți indica Latin-1 pentru XHTML:

```
<!ENTITY % ISOlat1 PUBLIC "-//W3C//ENTITIES Latin1//EN//XML"
                "http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent">>
%ISOlat1;
```

Cel la care tocmai ne-am uitat este definit ca

```
<!ENTITY lsquo "&#8216;"> <!-- left single quotation mark, U+2018 ISOnum -->
```

în interiorul setului de caractere speciale și este apelat astfel:

```
<!ENTITY % HTMLspecial PUBLIC "-//W3C//ENTITIES Special//EN//HTML"
                "http://www.w3.org/TR/xhtml1/DTD/xhtml-special.ent">>
%HTMLspecial;
```

Dacă veți face vreodată și ceva matematică, atunci veți avea nevoie și de acesta:

```
<!ENTITY % HTMLsymbol PUBLIC "-//W3C//ENTITIES Symbols//EN//HTML"
                "http://www.w3.org/TR/xhtml1/DTD/xhtml-symbol.ent">
%HTMLsymbol;
```

care conține lucruri ca

```
<!ENTITY cong     "&#8773;"> <!-- approximately equal to, U+2245 ISOtech -->
<!ENTITY asymp    "&#8776;"> <!-- almost equal to = asymptotic to, U+2248 ISOamsr -->
<!ENTITY ne       "&#8800;"> <!-- not equal to, U+2260 ISOtech -->
<!ENTITY equiv    "&#8801;"> <!-- identical to, U+2261 ISOtech -->
<!ENTITY le       "&#8804;"> <!-- less-than or equal to, U+2264 ISOtech -->
<!ENTITY ge       "&#8805;"> <!-- greater-than or equal to, U+2265 ISOtech -->
<!ENTITY sub      "&#8834;"> <!-- subset of, U+2282 ISOtech -->
<!ENTITY sup      "&#8835;"> <!-- superset of, U+2283 ISOtech -->
```

Aceste simboluri sunt utilizate de obicei în domeniile matematice și științifice. Acest set de simboluri conține de asemenea și litere grecești și alte mărunțișuri care să ușureze introducerea de conținut matematic în paginile dumneavoastră.

O metodă bună de a permite utilizarea ușoară a DTD-urilor modulare este de a furniza racorduri în document, care pot fi utilizate pentru a extinde aptitudinile DTD-ului atunci când este nevoie și a le ignora în rest.

Ca un exemplu, un DTD poate să înceapă cu entități care să indice la fișiere nul, astfel:

```
<!ENTITY % header SYSTEM "nullfile.dtd">
%header;
<!ENTITY % footer SYSTEM "nullfile.dtd">
%footer;
...
```


Atunci când doriți să *utilizați* efectiv informația antet (header) și informația subsol (footer), puteți să suprascrieți aceste valori cu unele care fac efectiv ceva, de genul:

```
<!DOCTYPE article SYSTEM "article.dtd" [
<!ENTITY % header SYSTEM "header.dtd">
<!ENTITY % footer SYSTEM "footer.dtd">
...
]>
```

În capitolul următor veți cerceta o altă metodă de a insera etichete noi, sau ramuri întregi.

Entități parametru

O *entitate parametru* este o metodă de a stoca date (sau de a indica spre ele pentru o utilizare ulterioară.

Entitățile parametru sunt valide numai în interiorul unui DTD și au comportamente diferite în submulțimea internă a DTD-ului și în submulțimea externă. Acest comportament asimetric a fost ales pentru a ușura analiza unui DTD intern de către un procesor XML care nu validează.

Notă După ce ați petrecut atât timp clarificând diferitele moduri în care puteți utiliza entitățile parametru pentru a vă face viața mai ușoară, este oarecum ironic că majoritatea propunerilor de scheme XML aproape că renunță la entitățile parametru în favoarea unei sintaxe gen XML, în opoziție cu una gen DTD. Așa se întâmplă...

Într-o entitate parametru puteți să puneți oricâte lucruri doriți și să le utilizați așa cum doriți. O utilizare tipică este de a citi fișiere externe dintr-un DTD, fie din submulțimea externă, fie din cea internă. O altă utilizare obișnuită este de a stoca bucăți integrale din părți de marcaje pentru a putea fi utilizate într-un mod mnemonic ca o prescurtare pentru expresii complexe. Pentru bucățile de marcaj care sunt utilizate iar și iar, câștigul în claritate poate fi enorm.

Poate veți dori să parcurgeți Capitolul 4, „Extinderea definiției tipului documentului prin modificări locale", pentru un exemplu complet de lucru cu entități parametru. Exemplul următor extinde un element definit, folosind entități parametru deoarece o copie a sa a fost făcută în submulțimea DTD-ului intern:

```
<!ELEMENT blink %Flow;>
<!ATTLIST blink
  %attrs;
>
```

Codul anterior, cu două entități parametru în două locații, a fost extins printr-o serie de căi ocolite la acesta:

```
<!ELEMENT blink (#PCDATA | p | h1 | h2 | h3 | h4 | h5 | h6 | div | ul | ol | dl |
menu | dir | pre | hr | blockquote | address | center | noframes | isindex |
fieldset | table | form | a | br | span | bdo | object | applet | img | map |
iframe | tt | i | b | big | small | u | s | strike |font | basefont | em | strong
| dfn | code | q | sub | sup | samp | kbd | var | cite | abbr | acronym | input |
select | textarea | label | button | ins | del | script | noscript)* >
<!ATTLIST blink
```

```
  id          ID        #IMPLIED
  class       CDATA     #IMPLIED
  style       CDATA     #IMPLIED
  title       CDATA     #IMPLIED
  lang        NMTOKEN   #IMPLIED
  xml:lang    NMTOKEN   #IMPLIED
  dir         (ltr|rtl) #IMPLIED
  onclick     CDATA     #IMPLIED
  ondblclick  CDATA     #IMPLIED
  onmousedown CDATA     #IMPLIED
  onmouseup   CDATA     #IMPLIED
  onmouseover CDATA     #IMPLIED
  onmousemove CDATA     #IMPLIED
  onmouseout  CDATA     #IMPLIED
  onkeypress  CDATA     #IMPLIED
  onkeydown   CDATA     #IMPLIED
  onkeyup     CDATA     #IMPLIED
>
```

Această extindere este în mod evident mai greu de citit decât echivalentul cu entități parametru, mai ales în cazul în care această construcție este utilizată în mai multe locuri.

Tabelul de entități W3C

Tabelul 3.2 explică pe scurt comportamentul entităților. Chiar dacă de fapt nu este totul atât de clar, un tabel asemănător este utilizat în Recomandarea XML 1.0 a W3C în încercarea de a clarifica regulile cărora li se supun entitățile în diverse contexte și este reprodus sau parafrazat aproape peste tot:

Tabelul 3.2 Comportamentele tipurilor de entități XML 1.0

Tipul entității	Parametru	Generale interne	Generale externe analizate	Neanalizate	Caracter
Referință în conținut	Nu este recunoscut	Incluse	Incluse dacă se validează	Interzise	Incluse
Referință în valoarea unui atribut	Nu este recunoscut	Incluse în valori literale	Interzise	Interzise	Incluse
Apariții ca valoare a unui atribut	Nu este recunoscut	Interzise	Interzise	Notificate	Nu este recunoscut
Referință în valoarea unei entități	Incluse în valori literale	Derivate	Derivate	Interzise	Incluse
Referință în DTD	Incluse ca PE	Interzise	Interzise	Interzise	Interzise

Haideți să încercăm să clarificăm această picătură de informație. Prima linie a tabelului se referă la locațiile din documentul în sine. A doua și a treia linie se referă la locații fie din document, fie din DTD. Liniile finale se referă numai la locații din DTD. Tabelul începe să aibă sens numai atunci când realizați că informația prezentată este concisă. Există suficiente intrări care vă spun numai că un anumit tip de entitate poate să apară într-o singură locație. Mă întreb dacă tabelul este chiar atât de valoros, în ciuda faptului că oamenii îl *vor* copia sau parafraza. În Recomandarea XML 1.0 actuală, aproape toate informațiile reale despre tabel sunt trecute în textul care îl referă.

Haideți să revedem cele câteva fragmente de informație conținute în tabel, sub forma a șapte reguli simplificate:

- Entitățile parametru pot fi utilizate numai în DTD. În exteriorul DTD-ului, secvența care le apelează, `%name;`, este tratată ca text simplu.
- În interiorul unei declarații, entitățile parametru sunt extinse, dar numai pentru submulțimea externă.
- În exteriorul unei declarații, entitățile parametru pot fi utilizate numai pentru a insera declarații de marcare complete, punând înainte și după fiecare declarație câte un spațiu.
- Entitățile generale interne pot fi utilizate numai pentru a insera text și sunt *derivate* în DTD. Aceasta înseamnă că ele sunt recunoscute, iar numele lor este introdus într-un tabel de căutare, fără valoare curentă. Atunci când sunt declarate, intrarea de căutare are o valoare asociată ei. Aceasta este o metodă obscură de a spune că le puteți utiliza înainte de a fi declarate.
- Entitățile generale externe analizate sunt tratate ca și entitățile generale interne, singura excepție fiind aceea că nu le puteți referi în valoarea unui atribut. Aceasta se datorează dificultății manevrării codificărilor caracterelor în contextul unui atribut.
- Entitățile neanalizate pot fi utilizate numai ca un nume în declarația unei entități. În acest context nimic altceva nu este recunoscut, inclusiv entitățile caracter. Singura responsabilitate a procesorului XML este să informeze aplicația de asistență, declarată în declarația unei notații, că entitatea neanalizată trebuie prelucrată.
- Entitățile caracter sunt tratate ca și entitățile generale, singura excepție fiind aceea că ele sunt incluse imediat în toate cazurile. Cu toate că sunt universale, în majoritatea cazurilor ele ar trebui utilizate ca entități generale dintr-un set predefinit. Referințele numerice brute, cu toate că sunt permise, sunt aproape imposibil de citit și înțeles. Deci, mnemonica `≠` este mai ușor de identificat decât echivalentul, indiscutabil mai dificil de reținut, `≠`, cu toate că ambele referă simbolul „diferit de" (≠).

Entitățile parametru sunt atât de puternice, de fapt, încât standardul XML 1.0 are grijă să înfrâneze puterea lor limitându-le prima dată la DTD și apoi limitând utilizările cele mai complexe la subsetul extern al DTD-ului, unde procesoarele XML care nu validează nu se uită niciodată (sau rareori).

Sugestie de la LeeAnne.com Words to weave by

Dacă utilizați entități parametru pentru a stoca text, fiți atent deoarece procesoarele XML inserează câte un spațiu înainte și după ele. Aceasta se face pentru a descuraja oamenii să le utilizeze pentru a stoca blocuri mici de text din care apoi să formeze cuvinte (sau mai ales marcaje!).

Nevoia de entități parametru, după toate probabilitățile, va dispărea atunci când una sau mai multe propuneri de scheme XML vor câștiga teren în mintea utilizatorilor. Dar până

atunci va mai trece un timp, deoarece standardele care se duelează trebuie să se împace.

Modificarea unui DTD XML

Diverse mecanisme sunt încorporate în DTD-urile XML pentru a vă permite să modificați un DTD existent. Primul dintre acestea, marcajul condițional, trebuie încorporat în DTD atunci când se creează, dar după aceea puteți modifica din mers comportamentul său prin suprascrierea valorii uneia sau mai multor entități parametru. Celelalte mecanisme, descrise în secțiunile următoare, adăugarea de atribute sau elemente noi, necesită mai puțină planificare, dar efectele lor sunt invizibile pentru ceilalți utilizatori ai DTD-ului. Alegerea mecanismului corespunzător fiecărei situații este o decizie de proiectare.

Marcaje condiționale

Multe persoane nu văd rostul marcajelor condiționale. Totuși, această facilitate oferă proiectantului DTD-ului posibilitatea de a crea versiuni de „depanare" sau alternative ale DTD-ului, care rezidă în fișierul în sine, de unde nu se pot pierde și unde puteți încuraja persoanele care modifică orice să se asigure că manevrează și părțile condiționale.

Marcajele condiționale utilizează mecanismul obișnuit al entităților parametru pentru a declara articole particulare ca având valoarea `IGNORE` sau `INCLUDE`. Ele seamănă puțin cu secțiunile `CDATA`, singura excepție fiind aceea că acest conținut trebuie să fie un marcaj încadrat de secvențe condiționale speciale de pornire și de încheiere:

```
<!DOCTYPE book PUBLIC [...] [
<!ENTITY % debug 'INCLUDE' >
<!ENTITY % release 'IGNORE' >
]>
  <!ENTITY % debug 'IGNORE' >
  <!ENTITY % release 'INCLUDE' >
  <![%debug;[
  <!ELEMENT book (comments*, front, body, back?)>
  ]]>
  <![%release;[
  <!ELEMENT book (front, body, back?)>
  ]]>
```

Pentru a comuta înainte și înapoi între versiuni care permit și care nu permit anexarea de comentarii la carte, trebuie numai să anulați valorile entitate `debug` și `release` undeva în submulțimea internă a DTD-ului. Observați că șirurile `INCLUDE` și `IGNORE` sunt cuvinte cheie XML și trebuie scrise, după cum se vede, cu majuscule; oricum, pentru parametri puteți folosi orice nume adecvat. Observați, de asemenea, că `debug` este doar o valoare. Acest mecanism îl puteți utiliza pentru a construi variante DTD de o complexitate arbitrară, printr-o alegere rațională a secțiunilor care să fie în mod selectiv adăugate sau eliminate și prin crearea entităților parametru care să le selecteze sau să le ignore. Este puțin probabil ca acest instrument să fie utilizat de mulți proiectanți de DTD-uri pentru proiecte mici, acesta fiind proiectat în SGML pentru medii cu zeci de realizatori de coduri care lucrează mai multe luni sau chiar ani. Atunci când DTD-ul dumneavoastră are mii de linii, nu vreți să încurcați lucrurile cu modificări de depanare ad-hoc. Faptul că puteți utiliza acest mecanism în alte moduri este un bonus.

Utilizarea atributelor noi

Un atribut poate fi adăugat unui element în orice moment deoarece listele de atribute sunt concatenate, iar prima declarație a oricărui nume de atribut are prioritate față de orice declarație următoare. Dacă nu există copii, atunci atributul este adăugat la lista atributelor existente, sau se creează una nouă dacă atributele nu au fost declarate anterior. Capitolul 4 dezbate în detaliu acest subiect.

Trebuie doar să adăugați o declarație `ATTLIST` la submulțimea internă a DTD-ului sau, în mod echivalent, un fișier DTD extern care copiază declarația noului dumneavoastră atribut înainte de a-l copia corect în DTD. Aici, noi am adăugat la atributul `alt` un câmp necesar pentru o imagine, pentru a încuraja oamenii să îl folosească așa cum recomandă WAI:

```
<!DOCTYPE book [
<!ATTLIST image alt CDATA #REQUIRED>
]>
```

Utilizarea elementelor noi

Cu toate că poate fi făcută cu ușurință, adăugarea elementelor este puțin mai dificilă decât adăugarea atributelor. Trebuie să găsiți un loc pentru ele în DTD-ul existent și să parcurgeți un proces uneori lung pentru a le potrivi, dar în mare măsură este un lucru ușor. Capitolul 4 studiază în detaliu acest subiect. Acest exemplu extrem de simplificat utilizează submulțimea internă a DTD-ului pentru a adăuga un element nou, `postit`, cu un model de conținut simplu pentru DTD-ul `book`:

```
<!DOCTYPE book [
<!ELEMENT postit #PCDATA >
]>
```

Nu se definește un loc pentru element în modelele de conținut ale DTD-ului, și nu se definesc nici atributele pe care este posibil să le ia elementul, dar veți vedea în Capitolul 4 cum le puteți realiza pe amândouă.

DTD-uri XHTML

XHTML este o încercare de a copia, în limitele XML, funcționalitatea lui HTML 4.0. Cu toate că există diferențe subtile, este în mare măsură un succes. Documentele HTML validate pot fi convertite destul de ușor în documente XHTML valide.

În Capitolul 4, veți studia ideea modificării DTD-ului XHTML pentru a-l adapta cerințelor proprii, ceea ce vă oferă oportunitatea de a studia DTD-ul în detaliu. DTD-ul XHTML tranzitoriu se găsește la sfârșitul acestei cărți.

Studii de caz

În acest capitol ați studiat în detaliu DTD-urile XML. Ar trebui să înțelegeți cum sunt construite, atât ca întreg, cât și în parte. Fiecare declarație cu DTD-uri are o sintaxă exactă cu reguli care trebuie urmate, iar macro-facilitatea XML – entitățile parametru – are un set de reguli chiar mai stricte, care îi limitează utilizarea numai la submulțimea externă a DTD-ului.

Oricum, întreaga problemă a definirii și utilizării DTD-urilor este una spinoasă în XML. Majoritatea lucrurilor într-adevăr utile pe care ați vrea să le faceți, cum ar fi includerea într-un document de ilustrații grafice sau definirea de ID-uri în interiorul unui document astfel încât referințe stabile la anumite locații să poată fi indicate utilizând XPointer și XPath, necesită ca DTD-ul să fie analizat de un analizor care validează.

Analizoarele care validează au propriile lor probleme. Deoarece ele necesită ca documentul să fie citit complet înainte de afișarea oricărei părți din el, ele încetinesc încărcarea paginii, vizualizarea ei devenind iritantă, cu excepția cazului în care documentul se află pe hard-discul calculatorului pe care doriți să îl vizualizați. Deoarece este posibil ca DTD-ul în sine să nu fie disponibil la cerere, sau poate necesita terminarea unei secvențe de apel destul de lungi, vă puteți aștepta ca timpul de încărcare să varieze destul de mult, chiar și numai din această cauză.

Anumite browsere XML pot încerca să evite această problemă prin încorporarea informațiilor anumitor DTD-uri în codul lor, dar aceasta ridică o problemă de latență atunci când un DTD se modifică, indiferent de motiv. Cu toate că DTD-urile ar trebui să nu se modifice nici măcar la perioade geologice, nu există nici o metodă de a asigura acest lucru decât bunăvoința autorului. Pentru orice astfel de schemă care îmbunătățește timpul de încărcare a documentului prin ascunderea informațiilor în browser, există posibilitatea „validării" față de o copie locală incorectă a DTD-ului.

CAPITOLUL 4

Extinderea definiției tipului documentului prin modificări locale

În acest capitol

Extinderea unui DTD
Crearea extensiei unui DTD
Crearea unui DTD extensibil
Crearea unui DTD modular
Furnizarea de valori locale unui DTD extern
Utilizarea DTD-urilor XHTML pentru redarea documentelor XML în browserele moștenite
Studii de caz

Extinderea unui DTD

În Capitolul 3, „Definițiile tipurilor documentului XML", ați descoperit ce era un DTD și ați învățat despre posibilitatea de a-l extinde. Deoarece un DTD XML este baza schimburilor între mulți autori de documente sau între cei care creează automat documente, și deoarece DTD-ul actual poate fi arhivat într-o locație asupra căreia nu aveți control, deseori este imposibil să modificați DTD-ul fără acordul tuturor utilizatorilor DTD-ului. Dar uneori există comunități locale sau grupuri de interes pentru care DTD-ul nu conține unele caracteristici care ar facilita comunicația, în ciuda faptului că alții nu ar vedea motive pentru care să îl modifice, deci este imposibil ca toată lumea să cadă de acord asupra modificărilor care ar trebui făcute. În continuare sunt prezentate trei metode prin care un DTD poate fi extins local fără a modifica submulțimea externă a DTD-ului:

- În submulțimea internă a unui document DTD puteți anula valorile `ENTITY` din submulțimea externă; această metodă este subiectul unor constrângeri destul de severe.
- În submulțimea internă a unui document DTD puteți anula liber sau puteți extinde declarațiile `ATTLIST` din submulțimea externă.
- Puteți adăuga elemente și orice alte declarații atât timp cât ele nu există deja în submulțimea externă XML, care, de cele mai multe ori, este DTD-ul în sine. Acest lucru poate fi foarte util dacă DTD-ul extern îl permite.

Atenţie Dacă utilizaţi orice entitate parametru externă care nu este citită de un procesor care nu validează, atunci toate entităţile şi declaraţiile de liste de atribute care urmează sunt ignorate. Cu toate că este adecvat, acest comportament poate să vă deruteze.

În multe situaţii, aceste trei opţiuni vă oferă o flexibilitate considerabilă şi vă permit să extindeţi destul de uşor un DTD, cu condiţia ca proiectantul DTD-ului să vă permită acest lucru. V-ar ajuta mult ca DTD-ul să fi fost proiectat pentru o extensibilitate uşoară şi există diverse tehnici pe care le puteţi utiliza pentru a controla exact care părţi ale DTD-ului pot fi modificate şi care nu, dacă aveţi plăcerea deosebită de a proiecta propriul dumneavoastră DTD.

Adăugarea unui atribut nou la un element existent

Cea mai simplă metodă de a adăuga un nou volum de informaţii este de a-l plasa într-un atribut. Deoarece un ATTLIST poate fi ascuns sau extins cu uşurinţă în submulţimea locală a DTD-ului, puteţi să adăugaţi un atribut la un element sau să îi modificaţi definiţia doar prin redefinirea ATTLIST-ului asociat lui. Pentru a evita conflictele, XML fuzionează valorile din ATTLIST-uri care se referă la acelaşi ELEMENT. Dacă un atribut are exact acelaşi nume, atunci prima referinţă (adică cea din submulţimea internă a DTD-ului) este păstrată, chiar dacă un ATTLIST ulterior utilizează o altă definiţie. Aceasta este o problemă cu instrumentele exclusiv-SGML mai vechi, deoarece SGML nu permitea redefinirea unui ATTLIST până când XML a impus acest lucru. Dar toate instrumentele SGML/XML moderne ar trebui să accepte această utilizare.

Ca un exemplu, haideţi să începem cu următorul fragment de DTD, numit book.dtd:

```
<!ELEMENT book (title, … ) >
<!ELEMENT title (#PCDATA) >
<!1ATTLIST title author CDATA #REQUIRED
                 titletype #CDATA "Fiction"
                 …
                 >
…
```

Acesta trebuie să fie modificat prin adăugarea la descrierea formală a titlului cărţii a unui atribut referitor la limbă şi modificarea categoriei titletype (tip titlu) prestabilite din Fiction (ficţiune) în NonFiction (non-ficţiune). De obicei, la submulţimea internă trebuie adăugată o singură instrucţiune, noul ATTLIST. Acesta va fi adăugat la documentul XML însuşi în submulţimea internă a DTD-ului, astfel:

```
<!DOCTYPE book [
<!ATTLIST title language #CDATA "German"
       titletype #CDATA "NonFiction" >
]>
```

Sugestie de la LeeAnne.com Words to weave by

XML permite ca un ATTLIST să fie redefinit de câte ori doriţi, dar prima instanţă a oricărui atribut îl defineşte în mod permanent. SGML a fost redefinit doar recent pentru a fi conform cu acest comportament, deci instrumentele SGML mai vechi pot lansa o excepţie atunci când sunt întâlnite aceste redefiniri multiple.

În utilizare, atributul nou şi valorile prestabilite sunt adăugate elementului title (titlu). Deoarece aproximativ tot ce puteţi face cu conţinutul unui ELEMENT se poate face şi cu atributul unui ELEMENT, puteţi adăuga destul de multe la un DTD prin această metodă, atât timp cât acolo există un element la care doriţi să adăugaţi datele dumneavoastră.

Adăugarea unui element nou

Adăugarea de elemente noi este mai delicată. Simplul fapt că puteţi defini un ELEMENT nou nu presupune neapărat că există un loc potrivit pentru a-l plasa în structura documentului ca întreg. Şi pentru a înrăutăţi şi mai mult lucrurile, mecanismul prin care este posibil să modificaţi cu uşurinţă sau să extindeţi un DTD extern – submulţimea internă – a fost sabotat de către proiectanţii XML, pentru a împiedica utilizarea de către dumneavoastră a entităţilor parametru în orice alt loc decât în nivelul superior. Cu alte cuvinte, codul

```
<!ENTITY % mizerie "gunoi">
%mizerie;
```

este legal în submulţimea internă, în timp ce mult mai valorosul

```
<!ENTITY % util "pretios">
<!ELEMENT %util; >
```

nu este. Aceasta înseamnă că nu vă puteţi uşura viaţa prin utilizarea acelorași notaţii prescurtate pe care le puteaţi utiliza în submulţimea externă a DTD-ului. Trebuie să extindeţi totul pentru a elimina orice referinţă parametru din interiorul altor elemente. Aceasta face programarea un pic mai uşoară pentru persoanele care doresc să analizeze submulţimea internă şi un pic mai dificilă pentru persoanele care doresc doar să lucreze cu ea.

Utilizarea entităţilor parametru existente pentru a modifica DTD-ul

Aceasta este o situaţie care are mare nevoie de un comportament orientat spre obiecte, iar XML are un mecanism care permite o moştenire limitată – parametrul ENTITY. În DTD trebuie să existe entităţi parametru care să poată fi anulate înainte ca dumneavoastră să puteţi extinde DTD-ul în această manieră.

Exemplul pentru acest proces utilizează DTD-ul XHTML 1.0 Transitional DTD Draft, dezvoltat de W3C, deoarece a fost proiectat pentru a putea fi extins cu uşurinţă. Primul lucru care trebuie observat este că aproape totul este definit ca ENTITY înainte de a fi utilizat ca şi conţinut ELEMENT. Aceasta permite o flexibilitate maximă în tratarea submulţimii externe, mai mult sau mai puţin ca un obiect.

Punctele de inserare trebuie mutate în submulţimea internă a DTD-ului, iar toate referinţele parametru ENTITY trebuie extinse la echivalentele lor, ceea ce poate fi plictisitor, dar poate fi făcut fără un efort excesiv. Cu suficientă atenţie, acest lucru poate fi făcut chiar şi în cele mai complexe instanţe, presupunând că entitatea este utilizată într-o manieră modulară. Dacă entitatea este utilizată în mai multe locuri, fiecare utilizare trebuie cercetată cu atenţie pentru a vă asigura că modificările dorite nu strică în mod neintenţionat altceva.

Ca un exemplu practic ne vom imagina că, în timp ce citiţi DTD-ul XHTML 1.0 Transitional DTD, observaţi că W3C nu a trecut eticheta dumneavoastră HTML preferată – legendara <BLINK>. Realizând repede că temeliile civilizaţiei se vor zgudui iar pros-

peritatea afacerilor se va prăbuși dacă nu modificați această situație pentru fișierele dumneavoastră locale, care utilizează intens <BLINK>, începeți să cercetați DTD-ul, care poate fi găsit în întregime în Anexa A, „Referințe XML/XHTML".

Notă Înainte de a scrie cuvinte aspre a căror expediere o veți regreta ziua următoare, trebuie să spun că sunt conștientă că nici o persoană normală nu va dori să facă asta. Eticheta <BLINK> a fost aleasă ca exemplu deoarece toată lumea este obișnuită cu comportamentul dorit și este și o problemă destul de complexă. Ea ilustrează dificultatea efectivă a inserării unei etichete, care face mai mult decât adăugarea unei CDATA mici și neînsemnate într-un mediu parametrizat.

Deoarece <BLINK> poate să apară atât in-line cât și ca etichetă de bloc, este logic să se înceapă căutarea într-o locație care acceptă etichete asemănătoare, cum ar fi <ins> și <del>. Faptul că aceste două etichete nu sunt limitate la nici un tip poate fi observat chiar din inspectarea DTD-ului, care este documentat într-un mod admirabil. Următorul cod este codul sursă efectiv al DTD-ului XHTML 1.0 DTD, care este utilizat ca un exemplu de DTD care nu poate fi modificat la sursa sa:

```
<!--============================ Inserted/Deleted Text ============================-->
<!--
  ins/del are allowed in block and inline content, but it's
  inappropriate to include block content within an ins element
  occurring in inline content.
-->
<!ELEMENT ins %Flow;>
<!ATTLIST ins
  %attrs;
  cite         %URI;          #IMPLIED
  datetime     %Datetime;     #IMPLIED
  >
<!ELEMENT del %Flow;>
<!ATTLIST del
  %attrs;
  cite         %URI;          #IMPLIED
  datetime     %Datetime;     #IMPLIED
  >
```

Avertismentul referitor la includerea unui bloc de conținut în acest gen de ELEMENT utilizat in-line reflectă un punct slab atât în XHTML 1.0, cât și în XML; în XML este dificil să excludeți conținut fără paliative obositoare. Autorii DTD-ului XHTML au hotărât să permită autorilor XHTML să își facă propria monitorizare și permit această ambiguitate particulară, chiar dacă este periculoasă.

Cea mai obișnuită metodă de a ocoli excluderile este de a crea un model de conținut special pentru un anumit element. Dar pentru a extinde excluderea asupra tuturor elementelor copil, trebuie ca fiecare element copil să fie redefinit cu propriul său nume. Această operație devine repede greoaie și este predispusă la erori, excepție făcând doar cazurile cele mai simple.

Atenție Excluderile sunt dificile în special la conversia unui DTD SGML la o versiune XML. Ele sunt utilizate din plin în majoritatea DTD-urilor SGML, iar debarasarea de ele sau ocolirea lor poate fi extrem de enervantă. În cel mai rău caz, este mai bine să recreați DTD-ul XML pornind de la zero.

Utilizând ca model definițiile ELEMENT și ATTLIST pentru <ins>, poate fi creat un model de definiție pentru <BLINK> care, eventual, poate fi plasat în submulțimea internă a DTD-ului, astfel:

```
<!--============================ Blinking Text ============================-->
<!--
  blink is allowed in block and inline content, but it's
  inappropriate to include block content within a blink element
  occurring in inline content.
-->
<!ELEMENT blink %Flow;>
<!ATTLIST blink
  %attrs;
>
```

<ins> este definit utilizând o entitate numită Flow, deci putem spera că se va strecura în elementul nou, deoarece entitățile sunt cheia extensibilității ușoare. În primul rând, entitatea Flow trebuie extinsă într-un mod curat. Referindu-ne la codul exemplu, %Flow este definită ca o combinație de mai multe tipuri de entități parametru și un tip de element ca acesta:

```
<!-- %Flow; mixes Block and Inline and is used for list items etc. -->
<!ENTITY % Flow "(#PCDATA | %block; | form | %inline; | %misc;)*">
```

După cum puteți vedea, entitatea Flow utilizează o serie de entități parametru pentru a-și defini modelul de conținut.

Sugestie de la LeeAnne.com Words to weave by — Cu toate că este obositoare, extinderea entităților parametru este un proces foarte mecanic. De fapt, el poate fi automatizat în mod profitabil utilizând un program simplu Java, sau chiar Perl.

Extinderea entităților parametru

Prezența entităților parametru în modelul nostru de conținut Flow ne creează o mică problemă deoarece noile entități parametru dezvăluite trebuie extinse înainte de a putea continua. La naiba! Amintiți-vă, entitățile parametru trebuie să fie definite *înainte* de a putea fi utilizate. Pe de altă parte, granulozitatea fină dezvăluită de acestea înseamnă și că există mai multe șanse de a găsi locația ideală pentru orice modificare dorită. Vom încerca să neglijăm faptul că, înainte de a încheia, fiecare entitate parametru vizibilă acum trebuie de asemenea extinsă, inclusiv cele din ATTLIST, fiecare având mai mulți strămoși. Of! Nici măcar Sisif rostogolind la nesfârșit bolovanul în sus pe munte, numai pentru a-l vedea rostogolindu-se înapoi, nu a avut o sarcină mai demoralizantă. Dar tot ce putem face este să încercăm.

După urmărirea întregului DTD și extinderea tuturor entităților care apar pe parcurs, este posibil să ajungem la o versiune potrivită a submulțimii interne a intrărilor noului DTD. Codul următor prezintă DTD-ul cu toate entitățile extinse complet:

```
<!--============================== Blinking Text ==============================-->
<!--
  blink is allowed in block and inline content, but it's
  inappropriate to include block content within a blink element
  occurring in inline content.
-->
<!ELEMENT blink (#PCDATA | p | h1 | h2 | h3 | h4 | h5 | h6 | div | ul | ol | dl |
menu | dir | pre | hr | blockquote | address | center | noframes | isindex |
fieldset | table | form | a | br | span | bdo | object | applet | img | map |
iframe | tt | i | b | big | small | u | s | strike |font | basefont | em | strong
| dfn | code | q | sub | sup | samp | kbd | var | cite | abbr | acronym | input |
select | textarea | label | button | ins | del | script | noscript)* >
<!ATTLIST blink
  id            ID          #IMPLIED
  class         CDATA       #IMPLIED
  style         CDATA       #IMPLIED
  title         CDATA       #IMPLIED
  lang          NMTOKEN     #IMPLIED
  xml:lang      NMTOKEN     #IMPLIED
  dir           (ltr|rtl)   #IMPLIED
  onclick       CDATA       #IMPLIED
  ondblclick    CDATA       #IMPLIED
  onmousedown   CDATA       #IMPLIED
  onmouseup     CDATA       #IMPLIED
  onmouseover   CDATA       #IMPLIED
  onmousemove   CDATA       #IMPLIED
  onmouseout    CDATA       #IMPLIED
  onkeypress    CDATA       #IMPLIED
  onkeydown     CDATA       #IMPLIED
  onkeyup       CDATA       #IMPLIED
>
```

Codul extins complet arată că el conține în interiorul noului model de conținut al etichetei `blink` majoritatea elementelor din XHTML, ceea ce pare corect deoarece în interiorul etichetei putem avea aproape orice. Lista imensă de elemente pare să fie un tribut adus nivelului de compresie și îmbunătățire în lizibilitate care poate fi obținut cu una sau două declarații `ENTITY`. Totuși, crearea sa poate fi puțin obositoare, iar citirea ei vă poate speria. Dar, pentru acest exemplu, va funcționa.

Sugestie de la LeeAnne.com Words to weave by

Dacă analizorul dumneavoastră XML care nu validează nu citește submulțimea internă a DTD-ului, așa cum ar trebui să o facă, puteți testa tentativa dumneavoastră de rezolvare prin inserarea ei chiar la începutul unei copii locale a DTD-ului extern. Deoarece submulțimea internă este citită înaintea celei externe, singurul fișier combinat își va apropia comportamentul de comportamentul standard XML 1.0.

În acest moment, sintaxa elementului nou poate fi verificată utilizând orice validator XML. Deoarece noul element `blink` nu a fost inserat într-un model de conținut, el nu va face de fapt nimic, dar validarea în acest moment poate fi o verificare utilă a progresului dumneavoastră de până acum.

Găsirea unui loc potrivit pentru elementul nou

Având codul nou, este mai ușor să căutați cel mai bun loc pentru a face cea de-a doua modificare, găsirea unui loc unde să inserați eticheta nouă într-un model de conținut existent. Eticheta este definită și se pare că poate conține toate elementele corespunzătoare, dar nu avem un loc unde să o plasăm. Ea trebuie inserată acolo unde toate elementele corespunzătoare o pot conține. `PCDATA` nu va face acest lucru. Deoarece `block`, `form` și `inline` par, în mod intuitiv, nepotrivite, putem să nu le cercetăm. Cel mai bun loc este în `misc`, adică exact locul unde se află `ins` și `del`:

```
<!ENTITY % misc "ins | del | script | noscript">
```

Aceasta pare să fie locul potrivit pentru elementele `ins` și `del` în modelele de conținut XHTML, iar noi știm *a priori* că ele au același comportament cu cel pe care dorim să îl aibă elementul `blink`, deoarece acestea influențează aspectul textului dintre etichetele de pornire și de încheiere ale elementului.

BIBL. CENTR. UNIV. „MIHAIL KOGĂLNICEANU" IAȘI INTRARE

Sugestie de la LeeAnne.com Words to weave by

De fapt, dacă extensia dumneavoastră este cât de cât asemănătoare cu un element existent în DTD, aproape întotdeauna merită să cercetați toți predecesorii acelui element înainte de a alege aleator un loc unde să îl plasați în documentul dumneavoastră.

Această concordanță face ca alegerea elementului `misc` ca locație pentru noul nostru element `blink` să fie și logică și convingătoare.

Încorporarea elementului nou în DTD-ul XHTML

Pentru a încorpora complet elementul nou în DTD-ul XHTML 1.0 Transitional DTD, valoarea existentă în submulțimea internă a DTD-ului trebuie înlocuită cu cea nouă:

```
<!ENTITY % misc "ins | del | blink | script | noscript">
```

Trebuie adăugată, de asemenea, și definiția extinsă a lui `ELEMENT` și a lui `ATTLIST`. Aceasta creează o înlocuire validă a DTD-ului care poate fi pusă în tabelul de căutare și ar trebui să funcționeze corect atunci când va fi combinată cu DTD-ul XHTML public.

Dar, înainte de combinarea DTD-urilor, mai trebuie realizată o operație. Fiecare DTD necesită un identificator de zonă de nume `xmlns` unic, care să permită agenților utilizator (browserelor) să identifice anumite DTD-uri. Deoarece DTD-ul XHTML este necesar pentru a defini parametrul `xmlns`, acesta trebuie de asemenea adăugat. Dar, deoarece DTD-ul XHTML este utilizat ca o bază, o entitate numită `XHTML-transitional-blink.ns` poate fi adăugată pentru a indica unde este localizat DTD-ul clipitor (blinking):

```
<!ENTITY % XHTML-transitional-blink.ns "http://www.leeanne.com/XML/blink.dtd">
```

Din păcate, DTD-ul XHTML tranzitoriu nu are definită o zonă de nume prestabilită prin acel nume de entitate, deci nu prea face altceva util în afară de documentarea intențiilor noastre. Dar pentru o documentație bună, mai trebuie spuse multe.

Iată codul complet:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/transitional.dtd" [
<!ENTITY % XHTML-transitional-blink.ns "http://www.leeanne.com/XML/blink.dtd">
<!ENTITY % misc "ins | del | blink | script | noscript">

<!--============================== Blinking Text ==============================-->
```

```
<!--
  blink is allowed in block and inline content, but it's
  inappropriate to include block content within a blink element
  occurring in inline content.
-->
<!ELEMENT blink (#PCDATA | p | h1 | h2 | h3 | h4 | h5 | h6 | div | ul | ol | dl |
menu | dir | pre | hr | blockquote | address | center | noframes | isindex |
fieldset | table | form | a | br | span | bdo | object | applet | img | map |
iframe | tt | i | b | big | small | u | s | strike |font | basefont | em | strong
| dfn | code | q | sub | sup | samp | kbd | var | cite | abbr | acronym | input |
select | textarea | label | button | ins | del | script | noscript)* >
<!ATTLIST blink
  id            ID          #IMPLIED
  class         CDATA       #IMPLIED
  style         CDATA       #IMPLIED
  title         CDATA       #IMPLIED
  lang          NMTOKEN     #IMPLIED
  xml:lang      NMTOKEN     #IMPLIED
  dir           (ltr|rtl)   #IMPLIED
  onclick       CDATA       #IMPLIED
  ondblclick    CDATA       #IMPLIED
  onmousedown   CDATA       #IMPLIED
  onmouseup     CDATA       #IMPLIED
  onmouseover   CDATA       #IMPLIED
  onmousemove   CDATA       #IMPLIED
  onmouseout    CDATA       #IMPLIED
  onkeypress    CDATA       #IMPLIED
  onkeydown     CDATA       #IMPLIED
  onkeyup       CDATA       #IMPLIED
>
]>
...
```

În listingul anterior, entitatea `misc` modificată se află în linia 4. Începutul noului element `blink` este la linia 11, iar noul `ATTLIST blink` începe la linia 18.

Şi iată cum arată codul complet într-un DTD extern:

```
<!ENTITY % XHTML-transitional-blink.ns "http://www.leeanne.com/XML/blink.dtd">
<!ENTITY % misc "ins | del | blink | script | noscript">
<!--============================== Blinking Text ==============================-->
<!--
  blink is allowed in block and inline content, but it's
  inappropriate to include block content within a blink element
  occurring in inline content.
-->
<!ELEMENT blink (#PCDATA | p | h1 | h2 | h3 | h4 | h5 | h6 | div | ul | ol | dl |
menu | dir | pre | hr | blockquote | address | center | noframes | isindex |
fieldset | table | form | a | br | span | bdo | object | applet | img | map |
iframe | tt | i | b | big | small | u | s | strike |font | basefont | em | strong
| dfn | code | q | sub | sup | samp | kbd | var | cite | abbr | acronym | input |
select | textarea | label | button | ins | del | script | noscript)* >
<!ATTLIST blink
  id            ID          #IMPLIED
  class         CDATA       #IMPLIED
```



```
  style         CDATA       #IMPLIED
  title         CDATA       #IMPLIED
  lang          NMTOKEN     #IMPLIED
  xml:lang      NMTOKEN     #IMPLIED
  dir           (ltr|rtl)   #IMPLIED
  onclick       CDATA       #IMPLIED
  ondblclick    CDATA       #IMPLIED
  onmousedown   CDATA       #IMPLIED
  onmouseup     CDATA       #IMPLIED
  onmouseover   CDATA       #IMPLIED
  onmousemove   CDATA       #IMPLIED
  onmouseout    CDATA       #IMPLIED
  onkeypress    CDATA       #IMPLIED
  onkeydown     CDATA       #IMPLIED
  onkeyup       CDATA       #IMPLIED
>

<!ENTITY % XHTML-Blink " PUBLIC „-//W3C//DTD XHTML 1.0 Transitional//EN"
                       "http://www.w3.org/TR/xhtml1/DTD/transitional.dtd" [
%XHTML-Blink;
```

Verificarea validităţii

Codul complet poate fi testat pentru a se vedea dacă este valid, atât timp cât există o conexiune la Web. Dacă nu există o conexiune, el poate fi testat cu fişierele locale – dar în acest fel se ratează scopul de a evita crearea de versiuni locale ale standardelor publice. Avantajul tuturor acestor lucruri este că fişierele clipitoare pot fi distribuite oricui, fără a convinge W3C (sau pe oricine altcineva) că `<blink>` este un lucru bun. Este clar pentru toată lumea că standardul este extins, nu modificat, iar izolarea elementelor extinse clarifică suficient modificările făcute. Această tehnică permite ca elemente noi să fie adăugate la aproape orice DTD. Figura 4.1 prezintă codul validat.

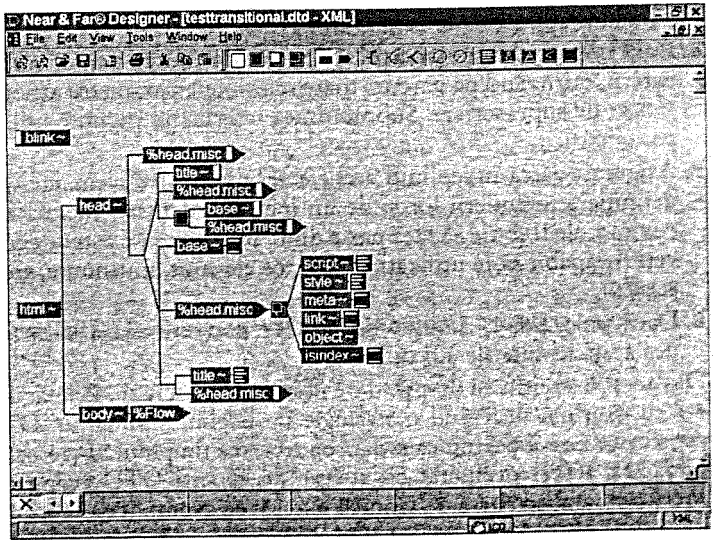


FIGURA 4.1

Acesta este DTD-ul validat utilizând Near&Far Designer™ de la Microstar Software Ltd. Deoarece acest produs este un designer DTD autonom şi nu un procesor XML, am utilizat strategia de testare explicată în nota din secţiunea anterioară „Extinderea entităţilor parametru", pentru a testa cu aproximaţie acest concept.

Notă Există zeci de instrumente de validare – unele mai bune decât altele. Dar Specificația XML 1.0 stabilește că orice lucru care pretinde că acceptă XML trebuie să accepte și submulțimea internă și să facă ceva inteligent cu aceasta. Un instrument care nu face acest lucru nu poate să pretindă cu adevărat că prelucrează fișiere XML. Anexa B conține o listă extinsă a instrumentelor de validare. Aceste instrumente sunt dezvoltate în mod constant pentru a face față competiției, deci înainte de a alege un instrument de validare ar trebui să îi cercetați posibilitățile actuale.

Crearea extensiei unui DTD

Procesul extinderii unui DTD prin modificarea submulțimii interne are avantajul portabilității, dar are și câteva probleme de întreținere. Secțiunea următoare rezumă pașii acestui proces într-o listă ușor de reținut. Cu toate că fiecare sarcină este ușor diferită, există suficiente asemănări pentru ca o listă generală de pași să aibă sens.

Extinderea unui DTD este în primul rând un exercițiu strategic deoarece secvența exactă de pași se modifică la fiecare DTD țintă. Studierea unei anumite secvențe de pași, așa cum s-a făcut în exemplul anterior, poate să demonstreze numai procesul general implicat. Testul real de abilitate strategică este realizarea operațiilor într-o situație necunoscută.

Aceeași secvență generală de activități ar trebui urmată la fiecare adăugare a unui element sau atribut la un DTD existent. Lista următoare descrie pe scurt acești pași, dar o anumită situație vă poate determina să schimbați ordinea lor sau să petreceți mai mult sau mai puțin timp cu fiecare dintre ei.

1. Cercetați DTD-ul extern pentru a găsi entități parametru care pot fi modificate în submulțimea internă a DTD-ului.
2. Căutați în listele de atribute locații în care atribute noi sau modificate pot adăuga informații necesare sau pot aduce alte avantaje.
3. Elaborați în amănunt o abordare, alegând fie una dintre ele, fie ambele metode, pentru a include structura de care aveți nevoie.
4. Dacă metoda dumneavoastră are nevoie de entități parametru, întoarceți-vă la DTD-ul extern, extinzând pe parcurs toate referințele parametru atunci când inserați declarații `ENTITY` de suprascriere. Submulțimea internă nu permite utilizarea liberă a entităților parametru.
5. Asigurați-vă că îndepliniți ambele cerințe pentru elementele noi. Ele au nevoie de o definiție a includerii lor și de un loc în structura generală a documentului. Cu alte cuvinte, ele trebuie să facă parte dintr-un element existent deoarece, în caz contrar, nu veți fi capabil să le utilizați. Deoarece ele sunt nou-nouțe, trebuie să le declarați și ca `ELEMENT`.
6. Dacă un `ELEMENT`, `ATTLIST` sau `ENTITY` nou utilizează `NOTATION`, acea notație trebuie declarată înainte de a fi utilizată.

Din acest exercițiu se desprind de fapt două concluzii. Prima: un DTD poate fi modificat cu ușurință utilizând o strategie în detaliu; a doua: un DTD conține elemente de proiectare care îi permit ca o dată cu trecerea timpului să crească și să evolueze. DTD-ul XHTML a fost prelucrat cu atenție de experți DTD având mulți ani de experiență. Merită studiat ca sursă de tehnică bună și ca exemplu de meșteșug.

Din nou despre adăugarea unui element

Adăugarea unui element nu este posibilă întotdeauna. Uneori nu există definiții `ENTITY` care să poată fi suprascrise sau elemente localizate potrivit pentru ca dumneavoastră să le puteți adăuga atribute. Singura soluție posibilă în acest caz este să faceți o copie locală și să o modificați direct. Evident, dacă funcționalitatea și structura care trebuie adăugate sunt foarte îndepărtate de DTD-ul existent, poate că ar fi mai bine să construiți un DTD nou – unul proiectat astfel încât să se potrivească cerințelor specifice – în loc să încercați să transformați unul nepotrivit într-o improvizație.

Pentru a ilustra acest lucru, presupuneți că, în exemplul anterior, declarațiile `ENTITY` existente nu sunt prezente și că nu există elemente la care să adăugați atribute. Din punct de vedere logic, un atribut `blink` poate fi adăugat la orice etichetă afișabilă pentru a rezolva problema într-un mod grosolan. Dar aceasta nu îndeplinește cerința ca eticheta `<blink>` să fie disponibilă pentru o utilizare generală.

În schimb, întregul DTD trebuie copiat folosind orice editor de texte. Modificările ar trebui făcute în mod direct, substituind entitatea `misc` din original și inserând elementul `blink` într-un loc corespunzător – chiar după elementele `ins` și `del`. De exemplu:

```
<!ENTITY % misc "ins | del | blink | script | noscript">
```

și

```
<!--============================== Blinking Text ==============================-->
<!--
  blink is allowed in block and inline content, but it's
  inappropriate to include block content within a blink element
  occurring in inline content.
-->
<!ELEMENT blink %Flow;>
<!ATTLIST blink
  %attrs;
>
```

În acest mod, submulțimea externă completă arată cam așa:

```
...
<!-- these can occur at block or inline level -->
<!ENTITY % misc "ins | del | blink | script | noscript">
<!ENTITY % inline "a | %special; | %fontstyle; | %phrase; | %inline.forms;">
<!-- %Inline; covers inline or "text-level" elements -->
<!ENTITY % Inline "(#PCDATA | %inline; | %misc;)*">
...
<!--========================== Inserted/Deleted Text ==========================-->
<!--
  ins/del are allowed in block and inline content, but it's
  inappropriate to include block content within an ins element
  occurring in inline content.
-->
<!ELEMENT ins %Flow;>
<!ATTLIST ins
  %attrs;
  cite          %URI;      #IMPLIED
  datetime      %Datetime; #IMPLIED
  >
```

```
<!ELEMENT del %Flow;>
<!ATTLIST del
  %attrs;
  cite          %URI;      #IMPLIED
  datetime      %Datetime; #IMPLIED
  >

<!--================================ Blinking Text ================================-->
<!--
  blink is allowed in block and inline content, but it's
  inappropriate to include block content within a blink element
  occurring in inline content.
-->
<!ELEMENT blink %Flow;>
<!ATTLIST blink
  %attrs;
>
...
```

În listingul anterior, entitatea `misc` modificată se află în linia 3. Noul element `blink` se află în linia 33, iar noul `ATTLIST blink` începe la linia 34.

Sugestie de la LeeAnne.com Words to weave by

Cu toate că modificarea brutală a unui DTD nu este de dorit, din mai multe motive – responsabilitatea publicării DTD-ului dumneavoastră modificat într-un anumit fel şi a menţinerii caracterului său perpetuu nu sunt ultimele dintre ele – în unele cazuri este singura opţiune. Dar şi forţa brută are avantajele ei. Cu toate că există o problemă de întreţinere la un DTD diferit, simplitatea relativă de a fi capabil să utilizaţi entităţi parametru face această abordare mai curată. În locul harababurii create prin extinderea referinţelor entităţi, există un DTD perfect care arată ca şi când ar fi fost scris aşa chiar de la început.

Rezumatul procesului de modificare a unui DTD

Codul exemplu anterior al unui DTD este scurtat prin înlocuirea textului cu puncte de suspensie pentru a economisi spaţiu. Cu toate că este modificată o singură instrucţiune şi sunt adăugate două instrucţiuni, pentru a face modificările, DTD-ul extern este copiat complet şi înlocuit. Acum, singura problemă este că DTD-ul modificat trebuie făcut disponibil separat pe Web pentru ca oamenii să îl poată utiliza. De asemenea, de fiecare dată când DTD-ul original este schimbat, DTD-ul modificat trebuie schimbat şi el. Metodele mai curate prezentate aici sunt de departe preferabile, pe cât posibil.

O alternativă ar fi copierea submulţimii externe a DTD-ului în submulţimea internă şi rezolvarea tuturor entităţilor parametru prin înlocuirea manuală a textului parametrizat cu valorile literale corespunzătoare. Această metodă este mult prea complicată pentru a fi viabilă.

În ciuda faptului că orice DTD poate părea extrem de complex, de obicei el se va descompune într-un număr de porţiuni mai mici care sunt uşor de înţeles şi utilizat. Partea grea este să ghiciţi cum să utilizaţi elementul nou şi cum se va potrivi acesta cel mai bine în structura existentă a documentului. În acest mic exemplu, înţelegerea HTML a ajutat la accelerarea procesului, dar o abordare similară funcţionează la orice DTD.

Sugestie de la LeeAnne.com Words to weave by

Este foarte dificil să reconciliaţi zonele de nume din DTD-urile modulare fără a adăuga complexitatea zonelor de nume XML. Cu toate că zonele de nume vă permit să specificaţi cărui DTD îi aparţin anumite elemente, nimeni nu pretinde că ele sunt uşor de înţeles sau de utilizat.

Atunci când creaţi DTD-uri avansate pe care poate veţi dori să le utilizaţi într-o manieră modulară, numele unice sunt foarte oportune. Ele reprezintă un paliativ mai uşor de înţeles pentru mecanismul primar. Poate veţi dori să utilizaţi un prefix de identificare la fiecare element pentru a forţa unicitatea, păstrând lizibilitatea prin sufixele mnemonice. În acest fel puteţi spune atât de unde vine un element, cât şi ce face acesta.

Când fuzionaţi sau modificaţi DTD-uri, este un lucru obişnuit să întâlniţi conflicte între zone de nume (ceea ce, simplificat, înseamnă că acelaşi nume de element sau de atribut este utilizat pentru două sau mai multe DTD-uri). Deoarece numele DTD-urilor sunt alese de obicei în aşa fel încât să fie mnemonice, numerele de telefon sunt deseori numite `numarTelefon` sau ceva apropiat. Aceasta face conflictele mult mai probabile decât în limbajele în care numele de variabile sunt alese după alte reguli. Deci, uneori este necesar să identificaţi sursa reală a unui nume. Metoda oficială a W3C de identificare a zonelor de nume şi de fuzionare sigură a DTD-urilor este utilizarea zonelor de nume XML, discutate detaliat în Capitolul 6, „Zone de nume XML".

Această metodă are avantajul că funcţionează întotdeauna, dar prefixarea continuă necesară şi învăţarea regulilor care se aplică în fiecare situaţie fac ca ea să fie obositoare şi predispusă la erori în cazul editării manuale. De fapt, zonele de nume XML au fost proiectate pentru a corespunde prelucrării automate; întrebuinţarea umană şi lizibilitatea au fost cerinţe secundare.

Această reducere a importanţei întrebuinţării şi lizibilităţii în favoarea universalităţii este destul de potrivită, deoarece o pagină poate fi citită de milioane de ori, dar este creată o singură dată, în schimb, pentru operaţiile realizate o singură dată care nu cer utilizarea de DTD-uri preexistente, asigurarea unicităţii numelor poate reduce considerabil dificultatea editării.

Crearea unui DTD extensibil

Acum, o a treia tehnică, care în cazul etichetei `blink` ar fi început de la celălalt capăt, este de a defini ca entitate parametru în submulţimea externă a DTD-ului orice lucru a cărui modificare este permisă. Astfel, un utilizator poate modifica valorile prestabilite prin redefinirea entităţii în submulţimea internă a DTD-ului local sau într-un DTD extern care încorporează DTD-ul de bază.

Dacă stabilitatea absolută este necesară în fiecare zonă, atunci evitaţi utilizarea entităţilor parametru. Aceasta este o metodă ideală de a permite extensibilitatea, deoarece DTD-ul se citeşte uşor, are o modularitate şi o granulozitate bună şi este întotdeauna complet în sine, deci poate fi validat independent.

Sugestie de la LeeAnne.com Words to weave by

Există câteva deprinderi de proiectare care fac extinderea unui DTD aproape trivială. Utilizarea intensă a entităţilor parametru pentru a permite o granulozitate bună şi accesibilă la nivelul codului este de mare ajutor, aşa cum modularitatea este bună la nivelul proiectării.

Înainte de a elimina, de dragul stabilității, entitățile parametru, gândiți-vă totuși la lizibilitatea codului rezultat. Uneori are sens să tolerați posibilitatea ca altcineva să modifice o secțiune care dumneavoastră ați prefera să rămână nemodificată, deoarece entitățile parametru fac codul mai ușor de citit și de întreținut în plan intern, dar și de modificat în plan extern.

Crearea unui DTD modular

S-a menționat anterior că este posibilă utilizarea de prefixe unice pentru a evita conflictele în zonele de nume, dar aceasta este doar o soluție interimară până când zonele de nume XML vor fi implementate complet în agenții utilizator XML. Crearea unui DTD care poate fi utilizat de oricine ca parte a fragmentelor proprii de DTD – indiferent de suportul agentului – este posibilă numai dacă autorul controlează toate părțile procesului de creare a unui DTD. În orice situație dată, nu există un control garantat asupra întregului proces, iar zonele de nume XML sunt necesare pentru a accepta combinațiile de DTD-uri arbitrare.

Pentru implementarea numelor unice, pot fi aplicate un număr de strategii, dintre care amintim numai una. Haideți să zicem că doresc să creez un DTD care accesează un anumit tip de fișier de date, cum ar fi Tandem SQL. O abordare bună ar fi prefixarea fiecărui element cu literele „TSQL-", deci probabilitatea conflictului cu o altă zonă de nume este destul de mică, chiar dacă zona de nume este utilizată ulterior de un alt proiectant. Când suportul zonelor de nume va fi universal, acest pas va fi inutil, dar până atunci este un paliativ excelent.

El oferă, de asemenea, un indiciu atunci când un DTD se strică deoarece cineva a modificat declarațiile DTD externe. În loc să încercați să depistați un element cu nume obscur, despre care nimeni nu pare să știe ceva, eticheta `<TSQL-access-method>` oferă de fiecare dată o sugestie clară despre originile sale. Ca să fim cinstiți, și zonele de nume XML permit elemente lipsite de ambiguitate.

De obicei, orice subset de funcționalitate clar definit este un candidat pentru a fi izolat de restul structurii dacă subarborele este corect definit în cadrul întregului document. Modularitatea mai înseamnă și că echipe diferite pot fi puse să lucreze la părți diferite ale proiectului. Aceasta asigură, de asemenea, și o speranță mai mare a realizării sarcinilor la timp și conform cerințelor.

DTD-uri modulare

Generația actuală de DTD-uri XML a fost dezvoltată în mare măsură în izolare, fiecare furnizor și fiecare domeniu creând așa-numitele „standarde" fără a se gândi la modul în care vor interacționa acele DTD-uri cu alte DTD-uri, sau chiar dacă noul DTD este sau nu necesar. Cu toate că aceasta ajută la mulțumirea tuturor părților implicate, duce de asemenea la repetarea eforturilor pentru fiecare proiect nou.

Sugestie de la LeeAnne.com Words to weave by

Sindromul NIH (Not Invented Here – Nu a fost inventat aici) a dus la o muncă inutilă pentru mult mai mulți proiectanți decât vă puteți imagina. Aceasta înseamnă că proiectanții de software, și chiar companiile, se împotrivesc utilizării în codul lor a modulelor scrise de altcineva, chiar și atunci când codul „străin" a fost testat și îmbunătățit ani la rând. Chiar și o improvizație care a fost făcută la repezeală și nu a fost testată este mai bună decât încrederea în alt programator.

XML în sine este utilizat pe mai multe platforme decât s-au așteptat inițial proiectanții HTML-ului. Răspândirea tipică a DTD-urilor specializate pentru fiecare tip de dispozitiv a făcut aproape imposibil să se țină pasul chiar și cu DTD-urile disponibile. În multe situații, „noile" DTD-uri pentru dispozitive specializate sunt numai subseturi, sau subseturi logice, ale DTD-urilor existente, deși numele au fost probabil modificate în așa măsură încât nu mai pot fi recunoscute.

W3C este conștient de această tendință și încearcă să încurajeze dezvoltarea de DTD-uri modulare care pot fi extrase în mod economic pentru diferite tipuri de aplicații sau medii țintă. Prin modularizarea DTD-urilor, proiectanții pot lucra asemenea inginerilor adevărați, specificând părți dintr-un catalog în loc să construiască totul pornind de la zero. Prin concentrarea asupra funcționalității și cerințelor aplicației în loc să se reinventeze fiecare roată și șurub în fiecare aplicație, dezvoltarea este intensificată, iar detaliile plictisitoare de nivel-jos ale implementării pot fi eventual evitate sau reduse.

Modular XHTML, un standard aflat în stadiu de dezvoltare de către W3C, va adresa prezentarea de nivel-jos și capacitățile de machetare ale multor dispozitive și medii fizice. DTD-urile cu conținut modular vor permite dezvoltatorilor de conținut să utilizeze componente funcționale asemănătoare pentru a construi DTD-uri mai mari, adresate anumitor nevoi de stocare și de transfer de date. Deoarece XHTML este proiectat astfel încât să fie extensibil, DTD-ul poate fi extins cu ușurință pentru a implementa noi posibilități pe măsură ce sunt dezvoltate noi dispozitive.

Un DTD modular poate fi format din mulțimi de elemente, mulțimi de atribute și mulțimi de modele de conținut, fiind posibilă implementarea tuturor utilizând documente externe care pot fi combinate pentru a crea DTD-uri XML hibride. Parametrii sau alte entități de substituție pot (și probabil trebuie) să facă parte din orice DTD modular pentru a ușura gestionarea complexității, chiar în cazul celor mai modulare DTD-uri.

Pașii pentru crearea unui DTD modular sunt următorii:

1. Identificarea acelor părți din modelul documentului dumneavoastră care sunt într-adevăr independente de celelalte și pot fi utile altora dacă sunt izolate. Modelul CALS pentru tabele, din SGML, a fost utilizat ca bază atât pentru tabelele HTML cât și pentru cele XHTML, iar elementele de prezentare din XHTML pot fi utile în mod separat.
2. Dacă găsiți părți modulare, asigurați-vă că secțiunea modulară este scrisă în mod independent și că entitățile parametru și documentele externe partajate cu alte DTD-uri modulare sunt utilizate într-o manieră controlată.
3. Definiți toate modelele de conținut ale elementelor, grupările logice de atribute și grupările de documente importante în entități parametru, deoarece aceste structuri sunt accesate și redefinite cu ușurință din exteriorul DTD-ului, ușurând astfel extinderea ulterioară a acestuia. Exemple de grupări bune de atribute sunt grupul de atribute de internaționalizare (i18n) din XHTML, atributele de evenimente interactive și colecțiile comune de evenimente utilizate în multe elemente.
4. Faceți experimente cu modularitatea DTD-ului, rafinându-l dacă este necesar pentru a-i extinde utilitatea și puterea.

Ca în orice abordare strategică a proiectării, este mai important să fiți flexibil și să răspundeți la situația de moment decât să aveți o abordare a proiectării bazată pe rețete standard.

Furnizarea de valori locale unui DTD extern

O altă abordare a extinderii unui DTD este să facilitați acest lucru chiar de la început. Se pot insera locații într-un DTD extern, care vor fi completate din submulțimea internă. Deoarece XML nu permite suprascrierea cu valori a entităților ELEMENT sau NOTATION în submulțimea internă a DTD-ului, aceasta înseamnă că ele trebuie să fie omise din submulțimea externă. Iar aceasta înseamnă că definirea lor în submulțimea internă este obligatorie.

Atenție Ambele abordări care urmează prezintă riscuri și sunt puțin confuze. Pentru simplitate, prefer să fac DTD-uri mai elegante și să le extind cu subtilitate, dar acestea sunt variante realizate cu forță brută; în unele cazuri, puțin mai multă finețe poate fi foarte valoroasă.

În mod alternativ, marcajele condiționale pot fi utilizate pentru a permite ca submulțimea externă să interzică includerea unui ELEMENT. În acest fel poate fi creat un DTD care este valid în sine și permite în continuare utilizatorilor să adauge modificări locale.

Ca un exemplu al primei abordări, să considerăm următorul fragment de submulțime externă a unui DTD:

```
<!ELEMENT whitepaper (companylogo, companyinfo, text, …)
<!ELEMENT text (#PCDATA) >
…
```

iar apoi, următoarea submulțime internă:

```
<!DOCTYPE whitepaper [
<!NOTATION jpeg PUBLIC "ISO/IEC 0918:1983//NOTATION
                       Digital Compresion and
                       Encoding of Continuous-tone
                       Still Images (JPEG)//EN">
<!ELEMENT companylogo SYSTEM "companylogo.jpeg" NDATA jpeg>
<!ELEMENT companyinfo CDATA #REQUIRED>
]>
```

Informația locală va fi disponibilă pentru document, dar poate fi redefinită după dorință pentru companii diferite. Abordarea este destul de bună, dar supărătoare dacă uitați să adăugați informația necesară și nu puteți valida DTD-ul extern în sine. Cea de-a doua abordare va fi mai sigură, făcând marcajele să fie condiționale și permițând utilizatorilor să suprime valorile prestabilite și să le furnizeze în DTD-ul intern. Fragmentele următoare implementează această abordare, pentru început în submulțimea externă a DTD-ului:

```
<!ELEMENT whitepaper (companylogo, companyinfo, text, …)
<!ENTITY supressdefault "INCLUDE">
<![supressdefault[NOTATION jpeg PUBLIC "ISO/IEC 0918:1983//NOTATION
                                  Digital Compresion and
                                  Encoding of Continuous-tone
                                  Still Images (JPEG)//EN"]]>
<![supressdefault[ELEMENT companylogo SYSTEM "defaultlogo.jpeg" NDATA jpeg>]]>
<![supressdefault[ELEMENT companyinfo CDATA #REQUIRED>]]>
<!ELEMENT text (#PCDATA) >
…
```

iar apoi, în submulțimea internă a DTD-ului:


```
<!DOCTYPE whitepaper [
<!ENTITY supressdefault "IGNORE">
<!NOTATION jpeg PUBLIC "ISO/IEC 0918:1983//NOTATION
                       Digital Compresion and
                       Encoding of Continuous-tone
                       Still Images (JPEG)//EN">
<!ELEMENT companylogo SYSTEM "companylogo.jpeg" NDATA jpeg>
<!ELEMENT companyinfo CDATA #REQUIRED>
]>
```

Cele două tehnici nu sunt deloc elegante, cu toate că pot fi utile în anumite situații.

Utilizarea DTD-urilor XHTML pentru redarea documentelor XML în browserele moștenite

Documentele XML vizualizate în majoritatea browserelor moștenite nu sunt foarte lizibile. Este posibil ca textul să nu fie continuu, iar etichetele XML nu înseamnă nimic pentru majoritatea browserelor HTML moștenite.

Până când XML va fi suportat peste tot, o abordare pentru suport XML este îmbinarea etichetelor XML, bogate în conținut și recunoscând zone de nume, cu etichetele XHTML care descriu macheta fizică. Din punct de vedere teoretic, aceasta permite unei aplicații să le elimine, să le ignore sau să le prelucreze pe ambele. Modul exact de realizare a acestui lucru este o parte a propunerii XML Schema, prezentată în Capitolul 7, „Scheme XML". CSS, XSL sau o „joacă" ad-hoc (și periculoasă) cu metodele brevetate ale anumitor browsere pot permite, de asemenea, ca documentele XML să fie afișate într-un browser moștenit.

Ca un exemplu de metodă relativ sigură de a îngloba conținut XML în etichete XHTML, să considerăm următorul fragment:

```
<html xmlns:wa="URI">
<head><title>Worth Ambition</title></head>
<body>
<wa:book>
<h1><wa:title>Worth Ambition</wa:title></h1>
<h2><wa:author>Lee Anne Phillips</wa:author></h2>
  [...]
</wa:book>
</body>
</html>
```

Observați că formatarea realizată este minimă, numai cât să facă fragmentul mai lizibil, iar documentul să fie redat mult mai elegant dacă browserul suportă XML, dar o formatare măcar minimală este furnizată și în browserele moștenite. Astfel, utilizatorilor nu li se prezintă o bucată confuză de text.

Studii de caz

Realizarea de modificări locale la un DTD permite unor utilizatori să modifice un DTD astfel încât acesta să se potrivească cerințelor lor, fără a fi nevoie să-i convingă pe toți uti-

lizatorii DTD-ului să facă modificări și fără să renunțe la unele dintre avantajele unui DTD partajat.

Ca un exemplu de necesitate din lumea reală care ar putea încuraja dezvoltarea unei variante locale de DTD, gândiți-vă la un furnizor care pune la dispoziție cataloage întregi de la mai mulți angrosiști.

Acel furnizor poate să primească de la angrosist multe informații despre produse, în format XML, dar dorește de asemenea să folosească un număr de catalog uniform, astfel încât utilizatorii catalogului său on-line să poată comanda cu ușurință produse, fără să riște confuzia care ar rezulta dacă numerele de catalog ale angrosiștilor s-ar suprapune.

Prin adăugarea unui câmp nou cu numărul de catalog al furnizorului la DTD-ul fiecărui angrosist, informația adăugată poate fi ușor prezentată utilizatorului fără a fi nevoie de transformarea documentului fiecărui angrosist într-un format comun. De fapt, prin transcludere, furnizorul nici măcar nu trebuie să asigure spațiu pe Web pentru cataloage întregi, ci se poate baza pe documente care să conțină doar numărul din catalogul furnizorului și o legătură la descrierea elementului de pe situl angrosistului. Elementul transclus poate fi formatat printr-o foaie de stil locală CSS sau XSL pentru a asigura un aspect unitar al sitului furnizorului.

Crearea unui DTD pornind de la zero

CAPITOLUL 5

În acest capitol

- Specificarea unui DTD personalizat
- Construirea unui DTD
- Principiile proiectării DTD-urilor
- Tratarea problemelor de excludere
- Tratarea problemelor de recursivitate
- Depanarea unui DTD defect
- Realizarea unui DTD blindat
- Evitarea tehnicilor de codare înșelătoare
- Asigurarea compatibilității DTD-urilor
- Studii de caz

Specificarea unui DTD personalizat

Construirea unui DTD personalizat nu este un lucru ușor. Este destul de obișnuit ca DTD-uri complexe să necesite serviciile de proiectare și codare ale mai multor persoane, timp de mai multe luni. Și mai este necesară și întreținerea continuă, pe măsură ce descoperiți erori sau defecte de proiectare care necesită remediere sau o nouă proiectare. Într-adevăr, nu diferă de un plan de documentație, cu care este echivalent un DTD. La definirea unui DTD, stabiliți într-un mod exact ideile dumneavoastră despre aspectul documentului, iar orice dezvoltare a documentului va necesita actualizarea DTD-ului.

Ca un mic exemplu, să presupunem că proiectați o revistă. Știți că toate articolele au un titlu, dar este oare posibil ca un articol să aibă un titlu în cuprins și alt titlu la începutul articolului? În multe reviste este așa, deci în această situație doriți două câmpuri pentru titlu – unul mai lung, pentru începutul articolului, iar altul, probabil mai scurt, pentru pagina cuprinsului. De asemenea, poate veți dori să aveți sau să nu aveți subtitluri. Aceasta depinde în întregime de aspectul pe care doriți să îl aibă revista în ansamblu.

Spre deosebire de hârtie, care este infinit mai flexibilă, un DTD, asemenea unui plan de documentație formal, este o întrupare fixă a ideilor dumneavoastră despre aspectul unui document și necesită efort pentru a fi modificat. Aceasta este și o protecție importantă. Asigurând faptul că aceste elemente sunt avute în vedere pe parcursul fazei de creație, în cazul producerii sau rescrierii, munca depusă va fi mai mică. Dar dacă descoperiți ulte-

rior că există o situație la care nu v-ați gândit, de exemplu un articol cu doi autori în timp ce dumneavoastră ați preconizat doar unul singur, va trebui să vă întoarceți la DTD și să îl remediați.

La urma urmelor, vă puteți aștepta să petreceți tot atât timp cu proiectarea DTD-ului cât ați petrece cu definirea machetei și stilului unei reviste, sau a standardelor de codare pentru un proiect de programare sau orice alt efort organizațional care presupune un proces de standardizare. Trebuie să intervievați și să luați în considerare ideile și cunoștințele în domeniu ale celor interesați – inclusiv ale conducerii, utilizatorilor, colaboratorilor, experților în domeniu și ale tuturor celor implicați în orice activitate căreia îi este destinat documentul XML.

Acest capitol se concentrează asupra abilităților necesare pentru a transforma acel domeniu de cunoaștere colectiv într-un DTD care permite realizarea fiecărei operații necesare și care nu vă încurcă (prea mult) atunci când adăugați „ciudățenii" din exterior. O problemă imensă la conversia informațiilor de pe hârtie într-o formă înțeleasă de calculator este faptul că se descoperă că tot felul de informații și referințe încrucișate au fost introduse între linii și pe marginile formularului care a fost creat la început. Aveți nevoie de cunoștințe în domeniu pentru a lua în considerare acele note de pe margine și a vă da seama cu adevărat de structura bazei de date de pe hârtie. Trebuie să cunoașteți principiile DTD-urilor pentru a evita recursivitatea sau alte erori. Aveți nevoie, de asemenea, de cunoștințe în domeniu pentru a realiza proiectarea sau reproiectarea proceselor bazându-vă pe DTD-uri sau standarde de schimb electronic care există deja pentru domeniul sau profesia dumneavoastră.

Dacă vă faceți bine treaba, utilizând procese și tehnici moderne de analiză a afacerii, veți obține o platformă robustă și stabilă, pe care pot avea loc evoluții ulterioare fără a fi nevoie de transformări distrugătoare. Poate veți încorpora strategii de dezvoltare rapidă și testare orientate spre obiecte, cum ar fi cele susținute de Walker Royce, Grady Booch, Ed Yourdon și alții, pentru a itera prin ciclurile de proiectare, implementare și evaluare, implicând activ în întregul proces pe cei direct interesați.

Construirea unui DTD

Această secțiune prezintă pașii implicați în crearea unui DTD, începând cu proiectarea și terminând cu un DTD complet, pregătit pentru testare și aprobare.

În continuare, este prezentat un scenariu tipic pentru gestionarea unui proiect software:

1. **Proiectarea** – Folosirea cunoștințelor și a experienței, împreună cu colectarea de date specifice, pentru a realiza specificația și planul unui DTD.
2. **Implementarea** – Acționați treptat de la un capăt la celălalt al dezvoltării pentru a crea un DTD modular.
3. **Testarea** – Testați DTD-ul utilizând diverse instrumente de depanare și testare.
4. **Iterația** – Întoarceți-vă la pasul de proiectare dacă este nevoie, pentru dezvoltări suplimentare ale DTD-ului.
5. **Aprobarea** – Acceptați și livrați un produs DTD finalizat.

După cum puteți vedea, același plan general de dezvoltare se aplică și la dezvoltarea DTD-urilor, iar același gen de instrumente de gestionare a proiectelor pot fi utilizate pentru a gestiona ambele tipuri de dezvoltare.

Proiectarea DTD-ului: adunarea datelor

Fiecare proces de proiectare începe cu adunarea de informații despre arta predecesorilor. În lumea ingineriei, arta predecesorilor nu înseamnă picturi ale vechilor maeștri, ci mai degrabă produse deja proiectate și realizate care corespund criteriilor dumneavoastră de proiectare.

Indiferent de domeniul dumneavoastră, de la medicină la activitatea editorială, sunt șanse ca cineva să se fi gândit deja la nevoile oamenilor din acel domeniu pentru a-și face meseria și să fi concretizat acel gând într-un DTD sau într-un alt standard de schimb electronic. O parte a efortului dumneavoastră de proiectare ar trebui dedicată identificării și studierii artei predecesorilor și a materialului informativ.

Anexa B, „Instrumente pentru editarea și transformarea XML/XHTML", furnizează o listă cuprinzătoare a organizațiilor care creează și publică DTD-uri. Printre ele este posibil să fie una care să corespundă cerințelor dumneavoastră și care a primit deja acordul altor membri din domeniu, sau una care să se apropie de așteptările dumneavoastră. Este ridicol să petreceți timp pornind de la zero în dezvoltarea unui lucru care s-ar putea să existe deja. Inginerii adevărați nu pierd vremea.

Chiar dacă nu găsiți un DTD preexistent, poate să existe un alt domeniu care poate fi studiat pentru a compara ideile și deciziile de proiectare anterioare. Candidații pentru această artă a predecesorilor, pentru a utiliza un termen curent din lumea proprietății intelectuale, pot fi cărți, cărți albe, programe, aplicații cu baze de date, discipline științifice sau orice lucru care reprezintă efortul ingineriei de a înțelege și organiza un domeniu de activitate. Un exemplu ar putea fi căutarea unei relații între sistemul zecimal al lui Dewey și sistemul de cote bibliografice al Bibliotecii Congresului SUA, înainte de a vă apuca de crearea unui DTD destinat unui catalog de fișe pentru o bibliotecă. În plus, deoarece sistemul MARC este un limbaj important pentru schimbul de date între biblioteci, poate veți dori să vă asigurați că întregul sistem de documente poate genera și înțelege cu ușurință înregistrări binare MARC.

Această secțiune studiază un domeniu care este bine documentat – activitatea editorială. De fapt, există multe DTD-uri care se adresează acestui domeniu, deoarece editorii și autorii sunt foarte obișnuiți cu marcajele, iar primele limbaje de marcare au fost proiectate chiar pentru a ușura publicarea. Pentru scopurile acestui exemplu, rezerva de artă a predecesorilor este ignorată, deoarece sarcina s-ar fi transformat în simpla operație de alegere a DTD-ului deja existent pentru a-l utiliza.

Acest exemplu se bazează pe un articol de revistă, deoarece operația pare simplă și servește la prezentarea modului în care puteți potrivi DTD-ul creat în interiorul altuia – o revistă – pentru a demonstra modularitatea.

Proiectarea DTD-ului: colectarea cunoștințelor din domeniu

Începeți cu aranjarea cunoștințelor despre domeniu, indiferent în ce ordine:

- Un articol are un titlu.
- Un articol poate avea o dedicație.
- Un articol poate avea un rezumat.
- Un articol are un corp de text.
- Un articol poate avea o anexă sau o secțiune de note după corpul textului.
- Un articol poate avea asociate informații bibliografice, data publicării, deținătorul

dreptului de autor, clasificarea subiectului, autorul și așa mai departe.

- Un autor are un nume, o adresă, eventual o afiliere la o universitate sau la o instituție, un anumit titlu (rang), probabil opțional în această țară (SUA) deoarece suntem înclinați să fim neprotocolari, și alte informații.
- Un corp de text poate avea note de subsol și de final.
- Un corp de text poate avea ilustrații sau tabele.
- Un corp de text poate avea secțiuni.

Lista poate continua; după cum puteți vedea, suma cunoștințelor noastre din domeniu începe să devină impunătoare, chiar și pentru lucruri simple, cum ar fi un articol de revistă. Există multe alte subiecte care nu au fost încă menționate, cum ar fi limba, setul de caractere utilizat, diverse tipuri de accentuare sau tratare specială a textului, texte tăiate și așa mai departe. Acest exemplu utilizează un subset minim din toate cele care pot fi necesare pentru realizarea unui articol, pentru a păstra exemplul compact și lizibil.

Sfârșitul procesului de proiectare ar trebui să aibă ca rezultat o idee clară despre scopurile proiectului. Pentru simplitate, să presupunem următoarele trei obiective (într-o ordine aleatoare):

- DTD-ul ar trebui să fie capabil să creeze un articol simplu de revistă, încorporând cât mai mult posibil din cunoștințele din domeniu.
- DTD-ul ar trebui să fie modular și ușor de extins.
- DTD-ul ar trebui să aibă nume de elemente ușor de tradus în alte limbi pentru a fi portabil între autori de diferite naționalități.

Acum, gândiți-vă la conținutul unui DTD – structura documentului, convențiile de stocare și părțile descriptive cu care se pornește, dar și convenții de prescurtare pentru denumirea și utilizarea colecțiilor de informații ca și când ar fi un singur lucru, prelucrările externe necesare pentru anumite tipuri de date și alte informații decisive despre fiecare aspect al unei instanțe a documentului.

Implementarea DTD-ului: începând cu datele

Începeți cu definirea numelor elementelor – echivalentul definirii dicționarului de date într-un proiect de programare procedurală sau a unei liste de obiecte în echivalentul orientat spre obiecte.

Pentru moment, presupunem că acest DTD se va numi `article` (articol), dar plasarea acestei informații într-o entitate parametru face posibilă suprascrierea ei în viitor. De fapt, acest lucru ar trebui făcut cu fiecare nume de element. Ar fi comod dacă toate numele etichetelor ar putea fi traduse în alte limbi, de exemplu, prin suprascrierea acestor declarații de entități fără a deranja DTD-ul în sine. Această metodă permite autorilor să vadă câmpuri descriptive sugestive atunci când editează un document. De exemplu, „editor“ în limba germană este *Verleger*, iar un autor german se va simți mai bine completând un câmp de date etichetat în germană și nu în engleză.

Sugestie de la LeeAnne.com Words to weave by

Prin plasarea tuturor numelor vizibile în entități parametru, permiteți autorilor care utilizează alte limbi decât engleza să utilizeze cu ușurință DTD-ul dumneavoastră, prin suprascrierea lor în submulțimea internă – înlocuindu-le cu traducerile echivalente. Documentul rezultat poate fi utilizat fie în forma sa în alte aplicații de publicare, fie să fie convertit cu ușurință înapoi la versiunea engleză a DTD-ului, utilizând XSLT, un limbaj de transformare a documentelor gen XML, discutat în Capitolul 11, „SAX (The Simple API for XML) – API simplu pentru XML“, și în Capitolul 13, „Foi de stil în cascadă și XML/XHTML“.

Adunarea tuturor numelor elementelor într-un singur loc ușurează alegerea lor pe măsură ce elementele sunt proiectate. La sfârșitul procesului, ar trebui să aveți o listă de nume aflate la același nivel de redirecționare – entități parametru – care adună într-un singur loc toate elementele care vor fi proiectate. Ca în orice proiect modern de programare, implementați, de asemenea, o *convenție de denumire*, cu entități parametru (locații care vor fi completate cu numele elementelor) scrise cu majuscule, iar textul de înlocuire scris cu litere mici. Poate veți dori să alegeți o altă schemă, mult mai elaborată, dar este important să puteți spune doar aruncând o privire dacă lucrați cu un nume de etichetă prin indirectare. În acest mod este mai greu să greșiți.

Listingul următor prezintă numele elementelor pe care dorim să le utilizăm în DTD-ul nostru, definite ca entități parametru, ele fiind astfel ușor de modificat:

```
<!-- Element names -->
<!ENTITY % Doctype     "article">
<!ENTITY % Front       "front">
<!ENTITY % Body        "body">
<!ENTITY % Back        "back">
<!ENTITY % Title       "title">
<!ENTITY % Author      "author">
<!ENTITY % Date        "date">
<!ENTITY % Publisher   "publisher">
<!ENTITY % Copyright   "copyright">
<!ENTITY % Part        "part">
<!ENTITY % Paragraph   "paragraph">
<!ENTITY % Number      "number">
<!ENTITY % Notes       "notes">
<!ENTITY % Appendix    "appendix">
```

În cea mai mare parte am utilizat convenția că numele entităților parametru sunt chiar numele cu care dorim să apelăm elementele, singura deosebire fiind aceea că prima literă este majusculă. Cel puțin pentru vorbitorii de limbă engleză, aceasta va ușura apelul corect al numelui entității atunci când scriu codul DTD-ului.

Implementarea DTD-ului: definirea rădăcinii

Acum că numele elementelor sunt definite, puteți începe definirea modelelor de conținut ale elementelor, începând cu rădăcina și completând ramurile arborelui documentului pe măsură ce înaintați. Aceasta rezolvă în primul rând arhitectura generală și apoi completarea muchiilor pentru a rafina și a extinde modelul. Ar fi la fel de corect să începeți cu o ramură mică și ușor de înțeles, să completați celelalte ramuri și apoi să le alipiți la rădăcină. Depinde de modul dumneavoastră de gândire, de la imagine generală la detalii fine sau invers.

În timp ce studiați rădăcina, gândiți-vă și la metode de a face acest DTD robust și extensibil, încorporând în fiecare element cât mai multe idei cu putință.
În rădăcină, nivelul cel mai înalt al documentului, puneți accentul pe un material introductiv, apoi pe corpul articolului, iar apoi adăugați eventual un material de încheiere. Puneți și aceste trei elemente în entități parametru, atât pentru a păstra ceva flexibilitate în timp ce scrieți, cât și pentru a ușura reutilizarea acestui format de bază atunci când creați un alt tip de document scris având proprietăți asemănătoare. Deoarece nu v-ați hotărât încă asupra structurării elementelor, lăsați-le deocamdată ca entități parametru necompletate – aici, echivalentul racordului unui program – și le veți completa ulterior.
Următorul fragment de cod vă arată cum este parametrizată și apoi utilizată structura de bloc de bază:

```
<!ENTITY % block      "(%front;, %body;, %back;)">

<!-- Document Root -->
<!ELEMENT %Doctype; ( %block; )>
```

Observați că lista de elemente numită `block` (bloc) se află în interiorul unei perechi de paranteze. Deoarece aceasta este o listă ordonată, nu aveți neapărată nevoie de ele. De fapt, deoarece parantezele încadrează chiar entitatea bloc din rădăcină, aceste paranteze sunt redundante. Dar dacă în viitor va fi adăugată o listă de selectare, sau dacă o pereche de paranteze va fi cândva eliminată din greșeală, este posibil să fiți recunoscător parantezelor „redundante". Ele nu dăunează, iar uneori pot fi folositoare.

Sugestie de la LeeAnne.com Words to weave by

Ar trebui ca aproape întotdeauna să încadrați între paranteze fiecare listă de elemente de model de conținut, chiar dacă nu aveți efectiv nevoie de ele. Parantezele lipsă din jurul unei liste de selecție reprezintă o sursă obișnuită de erori greu de localizat.

Observați, de asemenea, că numele „blocului" nu poate fi modificat. Cu toate că s-ar putea să vă pară rău ulterior, pare improbabil că veți dori vreodată să schimbați atât de ușor o declarație a rădăcinii. Trebuie să stabiliți o limită în fiecare DTD. Extensibilitatea este un scop nobil, iar localizarea este un lucru valoros. Dar acest lucru minor este păstrat parțial în acest exemplu ca un model simbolic pentru toate îmbunătățirile posibile care pot fi făcute, dar nu vor fi făcute aici pentru păstrarea simplității și finalizarea la timp a produsului. Capitolul 4, „Extinderea definiției tipului documentului prin modificări locale", arată cât este de comod pentru testările și modificările ulterioare dacă structura DTD-ului este păstrată în entități parametru, o construcție macro specială care permite modularizarea părților componente ale unui DTD în reprezentări compacte ale părților complexe.
Acum puteți să începeți să detaliați informațiile prin descompunerea materialului introductiv și a materialului de încheiere în părți componente. Înainte de a avansa prea mult, etichetați secțiunile pentru a vă putea da seama unde vă aflați. Dacă tot suntem aici, desfacerea în secțiuni poate fi referită, oricum, prin indirectare, deci vom face o nouă declarație de bloc pentru a păstra totul la un loc. Ca întotdeauna, treceți toate încercările de definiții în entități parametru, în caz că veți dori să le modificați cândva. Este mai ușor să faceți acest lucru pe măsură ce scrieți codul DTD-ului decât să vă întoarceți și să căutați fiecare instanță a oricărui nume.

Cu toate că acest lucru poate fi făcut în pași, codul următor prezintă produsul final, pentru a economisi timp. Observați că părțile componente ale blocului sunt etichetate cu un comentariu de identificare și sunt împărțite în secțiuni. Iarăși, așa cum ați văzut la modificarea DTD-urilor, în Capitolul 4, aceasta ușurează găsirea secțiunilor căutate, dacă ulterior doriți să extindeți sau să modificați documentul.

```
<!-- Element names -->
<!ENTITY % Doctype    "article">
<!ENTITY % Front      "front">
<!ENTITY % Body       "body">
<!ENTITY % Back       "back">
<!ENTITY % Title      "title">
<!ENTITY % Author     "author">
<!ENTITY % Date       "date">
<!ENTITY % Publisher  "publisher">
<!ENTITY % Copyright  "copyright">
<!ENTITY % Part       "part">
<!ENTITY % Paragraph  "paragraph">
<!ENTITY % Number     "number">
<!ENTITY % Notes      "notes">
<!ENTITY % Appendix   "appendix">
<!-- Content substitutions -->
<!ENTITY % %Front;    "(%Title;, %Author;, %Date;, %Publisher;?, %Copyright;?)">
<!ENTITY % %Body;     "(%Part;+|%Paragraph;+)">
<!ENTITY % %Part;     "(%Number;?,%Title;?,%Paragraph;+)">
<!ENTITY % %Back;     "(%Notes;, %Appendix;*) | (%Appendix;*)">
<!ENTITY % block      "(%front;, %body;, %back;)">
<!-- Document Root -->
<!ELEMENT %Doctype; ( %block; )>
<!-- Document Front Matter -->
<!ELEMENT %Title; (#PCDATA)>
<!ELEMENT %Author; (#PCDATA)>
<!ELEMENT %Date; (#PCDATA)>
<!ELEMENT %Publisher; (#PCDATA)>
<!ELEMENT %Copyright; (#PCDATA)>
<!-- Document Body -->
<!ELEMENT %Part; (%part;)>
<!ELEMENT %Number; (#PCDATA)>
<!ELEMENT %Paragraph; (#PCDATA)>
<!-- Document Back Matter -->
<!ELEMENT %Notes; (%Paragraph;)+>
<!ELEMENT %Appendix; (%part;)>
```

Atunci când adăugați indicatori de apariții, definiți în Capitolul 3, „Definițiile tipurilor documentului XML", treceți-i cât mai aproape posibil de definiția terminală pentru a-i face mai ușor de înțeles.
Cu toate că nu este perfect – de exemplu, nu există nici o metodă de a face un element aldin în interiorul unui paragraf – acest cod pare suficient pentru a începe testarea. În mod intenționat am inserat o eroare în linia 20, pentru ca dumneavoastră să aveți de depistat ceva atunci când testați ulterior codul.

Testarea DTD-ului

Pentru a testa DTD-ul, încercați să îl deschideți în Near & Far Designer, un designer DTD care validează în momentul încărcării.

Atunci când încărcați DTD-ul, el va furniza un mesaj de eroare – după cum se vede în Figura 5.1. Codul următor reprezintă mesajul de eroare rezultat:

```
Opening XML file: "d:\w3c\myarticle.dtd"
ERROR line 23 , column 22 :delimiter "|" invalid: only delimiter ",", delimiter ")"
and token separators are allowed
File read unsuccessfully.
```

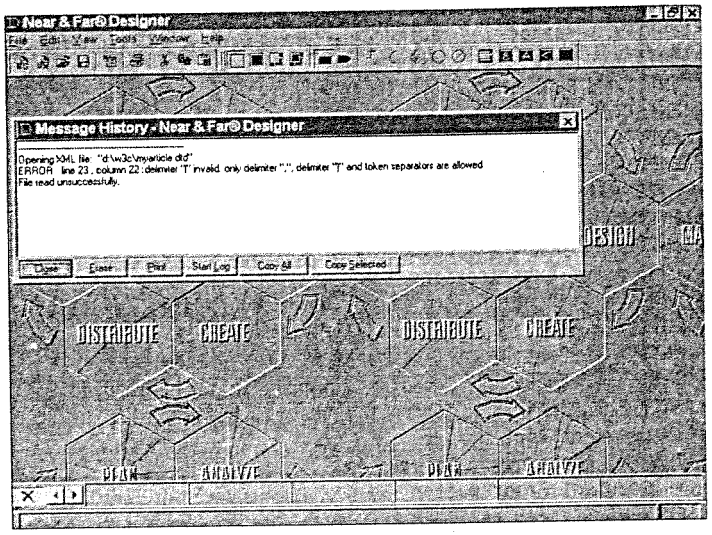


FIGURA 5.1
Lui Near & Far Designer™ nu îi place noul DTD.

Aceasta nu ne ajută, de fapt, prea mult, deoarece linia 23 arată astfel și *pare* să fie perfectă sau, cel puțin, atât de simplă încât ar fi greu să facem o greșeală:

```
<!ELEMENT %Doctype; ( %block; )>
```

Dar amintiți-vă că entitatea bloc va fi extinsă în toate acele entități parametru înainte ca linia să fie prelucrată. Trebuie să fie ceva în neregulă cu una dintre ele.

Încercați să vă uitați la lista extinderilor pentru a vedea care dintre ele are vreo eroare. În cuvintele `delimiter "|" invalid` există un indiciu. Uitați-vă prima dată la liniile care conțin un caracter `"|"`. Din fericire, există doar două:

```
<!ENTITY % %Body;     "(%Part;+|%Paragraph;+)">
<!ENTITY % %Back;     "(%Notes;, %Appendix;*) | (%Appendix;*)">
```

Studiați puțin și restul mesajului. `only delimiter ",", delimiter ")" and token separators are allowed`. Aceștia sunt delimitatorii unei liste ordonate atunci când nu există paranteze. Dacă tot nu vedeți nimic, iar într-un DTD complex poate fi mult mai greu, puteți să începeți să extindeți manual entitățile parametru, inserându-le una câte una în linia 23 până când ceva începe să aibă sens. În final, rămâneți cu același mesaj, dar care acum se referă la una dintre cele două instrucțiuni anterioare în care se află eroarea și indică mai precis locul problemei.

Ca și în Capitolul 4, extinderea entităților este un proces foarte plictisitor, deci pentru a scurta o poveste lungă, atunci când aveți o selecție, aceasta trebuie trecută între paranteze, care au fost uitate aici:

```
<!ENTITY % %Back;     "(%Notes;, %Appendix;*) | (%Appendix;*)">
```

Rescrieți-o astfel:

```
<!ENTITY % %Back;     "((%Notes;, %Appendix;*) | (%Appendix;*))">
```

De asemenea, rescrieți-o și în DTD-ul de test ca să îl puteți testa din nou. Acum, codul arată astfel:

```
<!-- Element names -->
<!ENTITY % Doctype    "article">
<!ENTITY % Front      "front">
<!ENTITY % Body       "body">
<!ENTITY % Back       "back">
<!ENTITY % Title      "title">
<!ENTITY % Author     "author">
<!ENTITY % Date       "date">
<!ENTITY % Publisher  "publisher">
<!ENTITY % Copyright  "copyright">
<!ENTITY % Part       "part">
<!ENTITY % Paragraph  "paragraph">
<!ENTITY % Number     "number">
<!ENTITY % Notes      "notes">
<!ENTITY % Appendix   "appendix">
<!-- Content substitutions -->
<!ENTITY % %Front;    "(%Title;, %Author;, %Date;, %Publisher;?, %Copyright;?)">
<!ENTITY % %Body;     "(%Part;+|%Paragraph;+)">
<!ENTITY % %Part;     "(%Number;?,%Title;?,%Paragraph;+)">
<!ENTITY % %Back;     "((%Notes;, %Appendix;*) | (%Appendix;*))">
<!ENTITY % block      "(%front;, %body;, %back;)">
<!-- Document Root -->
<!ELEMENT %Doctype; ( %block; )>
<!-- Document Front Matter -->
<!ELEMENT %Title; (#PCDATA)>
<!ELEMENT %Author; (#PCDATA)>
<!ELEMENT %Date; (#PCDATA)>
<!ELEMENT %Publisher; (#PCDATA)>
<!ELEMENT %Copyright; (#PCDATA)>
<!-- Document Body -->
<!ELEMENT %Part; (%part;)>
<!ELEMENT %Number; (#PCDATA)>
<!ELEMENT %Paragraph; (#PCDATA)>
<!-- Document Back Matter -->
<!ELEMENT %Notes; (%Paragraph;)+>
<!ELEMENT %Appendix; (%part;)>
```

Acum, când DTD-ul de test este citit de Near & Far Designer, vedeți ceva complet diferit. Extindeți toate grupurile pentru a-l face puțin mai interesant. Atunci când apare, sunt vizibile doar elementul rădăcină și grupul bloc. Figura 5.2 prezintă DTD-ul extins complet.

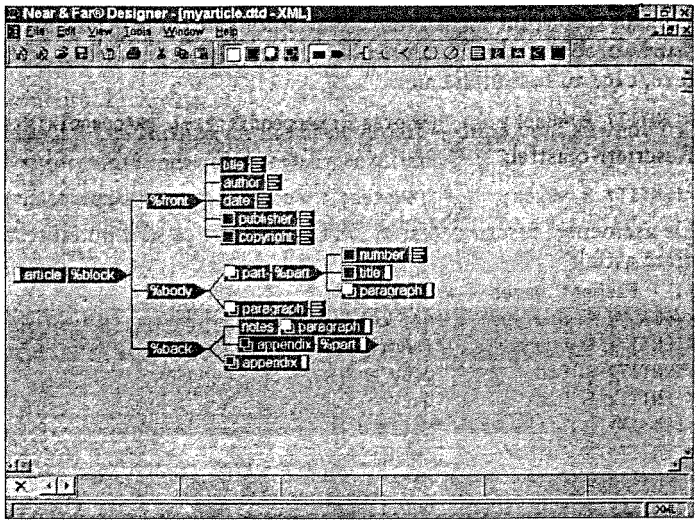


FIGURA 5.2
Acum puteți vedea de ce preferă oamenii instrumentele automatizate. Aici aveți o vizualizare grafică a structurii DTD-ului în Near & Far Designer. Este greu de imaginat cum ar putea fi îmbunătățit.

DTD-ul spune că trebuie să aveți titlu, autor și dată, urmate de date opționale privind editorul și drepturile de autor. Apoi, trebuie să aveți una sau mai multe părți, care au un număr opțional și un titlu opțional, urmate de unul sau mai multe paragrafe. Pare logic, deoarece un articol trebuie să aibă ceva în spatele unui titlu, altfel nimeni nu îl citește. În materialul de încheiere aveți fie note și anexe, fie numai anexe, fie nu aveți nimic.

După cum s-a menționat anterior, DTD-ului îi lipsesc încă multe lucruri care ar fi necesare într-un DTD complet – cum ar fi diferite tipuri de evidențiere, posibilitatea de a avea note de subsol și așa mai departe – dar acesta produce de fapt un rezultat care arată ca un articol real, chiar în forma sa actuală. Puteți acum să scrieți o foaie de codare, ceea ce nu ar trebui să fie atât de dificil, transferând autorilor sarcina scrierii articolelor. Autorii nu trebuie să cunoască nimic despre DTD-ul în sine, ci numai care etichete sunt disponibile și unde pot să apară.

Cu aceasta ne-am îndeplinit obiectivele proiectului nostru, care erau să facem DTD-ul capabil să reprezinte un articol de revistă simplu, să fie cât mai flexibil și extensibil cu putință și să permită ca fiecare nume de etichetă să poată fi suprascris pentru a-i ajuta pe autorii care utilizează o limbă străină. Ca și în Capitolul 4, entitățile parametru ușurează adăugarea unui element nou oriunde în arborele existent. De exemplu, dacă doriți să adăugați o casetă redacțională la materialul de încheiere, trebuie numai să conectați elementul nou, exact așa cum s-a făcut în Capitolul 4 cu elementul `blink`, să ajustați câteva declarații de entități și ați terminat.

Principiile proiectării DTD-urilor

Un DTD reprezintă un arbore cu o singură rădăcină și fără bucle sau ambiguități logice. În orice punct, acel arbore poate fi traversat utilizând o logică simplă. Prima activitate a efortului dumneavoastră de proiectare trebuie să fie identificarea și documentarea fiecărei structuri logice preexistente care se poate afla în documentul dumneavoastră țintă și pe care doriți să o definiți în echivalentul XML. Dacă această structură nu poate

fi transformată într-un tip de arbore, atunci poate va trebui să căutați în continuare, dar căutând suficient sunt șanse să găsiți o structură asemănătoare. Sunt puține domenii care nu pot fi organizate astfel, deoarece stă în firea omului să aranjeze lucrurile în categorii clare.

Poate că vă va fi de folos să scrieți încercările de clasificare, ideile, relațiile, câmpurile de date și orice altceva pe cartonașe de 3×5 inci, sau poate de 4×6 inci dacă aveți mult de scris, și să începeți să le așezați pe o masă sau pe podea. Dacă utilizați etichete adezive, puteți folosi un perete dacă nu vreți să vă târâți pe podea sau dacă proiectul dumneavoastră este vast. Poate veți vedea un model sau poate nu.

Atenție Ar trebui să rezistați tentației de a porni un instrument de proiectare automatizat și a începe să ajustați până când aveți pe hârtie un proiect care are sens. Instrumentele automatizate sunt lucruri fantastice și foarte valoroase pentru accelerarea etapelor finale ale dezvoltării DTD-urilor, dar trebuie să știți de la început unde doriți să ajungeți cu ele, altfel puteți fi dus de val și să pierdeți din vedere obiectivele inițiale.

Toată lumea se confruntă cu problema afișării unui proiect pe un monitor relativ mic. Nici măcar cu posibilitatea de a restrânge porțiuni din arbore și de a extinde altele nu puteți vedea deodată pădurea întreagă, ci numai mici pictograme reprezentând părți ale sale.

Ați văzut că și la micul DTD, vizualizarea extinsă a ocupat o jumătate de pagină în Near & Far Designer. În timp ce pictogramele devin din ce în ce mai mici pe măsură ce DTD-ul este extins, pierdeți cu repeziciune capacitatea de a-l vedea în întregime.

Există câteva preocupări cu privire la extensibilitate și la posibilitatea de localizare atunci când proiectăm acest DTD. Este posibil să nu aveți același set de probleme și să puteți evita un nivel de indirectare dacă autorii acceptă să utilizeze etichete în limba engleză – sau în franceză și așa mai departe.

Aceasta subliniază tocmai ce am afirmat mai înainte: proiectul trebuie să fie făcut primul, altfel nu veți ști ce faceți sau veți produce doar rebuturi.

Tratarea problemelor de excludere

Unele documente pot să definească structuri care nu trebuie să fie imbricate. Un exemplu HTML ar fi un element paragraf `<p>`, care nu poate fi conținut direct într-un alt element paragraf. Totuși, un paragraf poate fi imbricat într-un alt element la nivel de bloc, deci tot ce trebuie să faceți este să vă asigurați că modelul de conținut pentru un paragraf nu conține paragraful însuși și problema este rezolvată.

Dar unele elemente nu se pot imbrica, oricât de indirectă ar fi această imbricare. O excluziune simplă bazată pe modelul de conținut este posibil să nu funcționeze în această situație, deoarece părțile legitime ale modelului de conținut pot fi capabile să ducă elementul în alte contexte. Un exemplu ar fi elementul ancoră HTML `<a>`, care nu poate fi imbricat la nici un nivel. Un simplu model de conținut nu vă poate proteja împotriva acestui lucru. De fapt, în afara definirii unor nume duplicate pentru conținutul afectat, nu există nici un mod de a afirma în XML că un element nu poate fi propriul său descendent.

Trebuie să îi spuneți autorului dacă există astfel de restricții; XML nu are cum să oblige respectarea acestor restricții.

Tratarea problemelor de recursivitate

XML nu permite entităților analizate să fie propriii lor descendenți. Dacă astfel de entități ar fi permise s-ar crea o buclă infinită. Entitățile analizate recursive sunt ca și dicționarele care definesc un cuvânt în termenii altui cuvânt, iar pe celălalt cuvânt în termenii primului.

Imaginați-vă că găsiți o pereche de definiții care arată astfel: Zarf; obiectul metalic mic, ținut într-un finjan. Finjan; suport adesea decorativ sau suport pentru un zarf. Dicționarele sunt notorii pentru definițiile recursive – producții care depind unele de altele pentru a avea sens, într-o manieră circulară. Dacă știți ce este un zarf, atunci înțelesul lui finjan este clar și viceversa.

Notă Pentru a elimina orice frustrare posibilă, *zarf* este un cuvânt foarte obscur care se referă la o ceașcă de cafea turcească, mică, cu fundul rotunjit și fără toartă. *Finjan* este tot un cuvânt turcesc și se referă la suportul cești, care vă permite să o puneți jos fără să se răstoarne. Echivalentul european exact îl reprezintă acele cești de cafea de unică folosință, conice, de culoare albă, care încap într-un suport de plastic cafeniu. Cuvântul ZARF este foarte vestit în cercurile informatice deoarece a fost numele de cod al echipei de „tigri" din aviația americană, care a spart sistemul de securitate Multics. Ei au băut *multă* cafea.

Un DTD nu poate utiliza o logică circulară pentru a defini o structură arborescentă. Deci codul următor va produce o eroare:

```
<!ENTITY % zarf %finjan;>
<!ENTITY % finjan %zarf;>
```

Iar echivalentul său, utilizând entități analizate externe, este de asemenea greșit. Primul fișier se numește `zarf.xml`:

```
<!ENTITY zarf SYSTEM "http://www.leeanne.com/usingxml/finjan.xml">
&zarf;
```

iar cel de-al doilea se numește `finjan.xml`:

```
<!ENTITY finjan SYSTEM "http://www.leeanne.com/usingxml/zarf.xml">
&finjan;
```

Într-un analizor care validează, acestea ar produce o eroare fatală. Dar într-un analizor care nu validează, poate că nu.

XML nu are facilități sigure pentru prevenirea recursivității. În timp ce un analizor care validează poate că va detecta recursivități simple în entități externe analizate, un analizor care nu validează nu le va detecta. Chiar și unele analizoare care validează pot avea limite privind profunzimea entităților externe analizate pe care le vor cerceta pentru a asigura validitatea unei fișier local. Din punct de vedere teoretic, este posibil să creați un DTD utilizând recursivitatea indirectă, dar va dura o veșnicie (sau aproape o veșnicie) să determinați că DTD-ul nu va fi valid.

Depanarea unui DTD defect

Găsirea unei erori dintr-un DTD poate fi o sarcină obositoare. Găsirea de erori multiple este chiar mai grea. Majoritatea mediilor de editare și afișare vor refuza să încarce un

astfel de lucru diform și vor ieși cu un singur mesaj de eroare care poate că va identifica, sau poate că nu, numărul liniei la care procesorul își dă seama că a găsit o eroare. Punctul respectiv poate să nu fie punctul efectiv în care a fost făcută eroarea.

Cercetarea atentă a sursei și comparația cu mesajul de eroare pot ajuta la izolarea erorii. Totuși, multe instrumente au bug-uri care împiedică chiar și identificarea unor erori simple. Cel mai bine este să utilizați o diversitate de instrumente și să le încercați pe rând pe toate. Unele sunt mai bune decât altele pentru anumite erori. Într-o anumită situație, unele au mesaje de eroare mult mai informative. Dar nici unul nu este perfect.

Lucrurile care trebuie căutate sunt ghilimele lipsă, paranteze lipsă (ca în exemplul anterior), utilizarea unei entități parametru în submulțimea internă (este atât de tentantă), declarații scrise greșit (un alt motiv pentru a utiliza nume reale în locul codurilor arbitrare) sau omiterea faptului că notațiile și entitățile trebuie declarate înainte de a le putea utiliza.

Realizarea unui DTD blindat

DTD-urile sunt lucruri destul de robuste în sine. Cu atât de puține elemente, este greu să vă încurcați atunci când editați unul. Totuși, sunt câteva lucruri pe care le puteți face pentru a evita comiterea unor greșeli.

Începeți cu comentariile. Deși codul anterior a fost comentat minimal, într-un DTD de producție doriți să adăugați o prezentare generală a proiectului și unele indicii despre cum a fost construit DTD-ul și cum se pot face modificări simple. DTD-ul exemplu a fost proiectat astfel încât dumneavoastră să puteți schimba cu ușurință numele elementelor, deci ar trebui să explicați cum se procedează. Multe firme vă cer să vă identificați și în codul dumneavoastră. Nu numai că ar trebui să fiți mândru de proiectele dumneavoastră, dar dacă trebuie făcută vreo modificare, poate veți dori să fiți primul care încearcă. Codul următor prezintă modul în care puteți include informații despre cum se încadrează documentul într-o structură organizată, despre aspectul structurii sale generale și despre cine l-a creat, prin adăugarea de comentarii.

```
<!-- This is the article DTD and forms part     -->
<!-- The XYZ corporation document standards     -->
<!-- Typical invocation:                        -->
<!--     <!DOCTYPE article                      -->
<!--      PUBLIC "-//XYZ//Simple Article//EN"   -->
<!--             "article.dtd">                 -->
<!-- This DTD coded by Lee Anne Phillips 1999   -->
<!-- Element names                              -->
<!-- For easy localization, all Element names   -->
<!-- have been collected below as parameter     -->
<!-- entities. To localize this DTD, override   -->
<!-- each Name with a local equivalent name.    -->
<!-- All name containers are capitalized so     -->
<!-- they're easy to spot in the DTD but you    -->
<!-- don't have to touch them anywhere but      -->
<!-- here.                                      -->
<!ENTITY % Doctype      "article">
<!ENTITY % Front        "front">
<!ENTITY % Body         "body">
<!ENTITY % Back         "back">
```

```
<!ENTITY % Title        "title">
<!ENTITY % Author       "author">
<!ENTITY % Date         "date">
<!ENTITY % Publisher    "publisher">
<!ENTITY % Copyright    "copyright">
<!ENTITY % Part         "part">
<!ENTITY % Paragraph    "paragraph">
<!ENTITY % Number       "number">
<!ENTITY % Notes        "notes">
<!ENTITY % Appendix     "appendix">
```

Fiecare secțiune ar trebui extinsă pentru a fi mai clară, astfel:

```
<!-- Content substitutions -->
<!-- For easy modification, all content names -->
<!-- have been collected below as parameter   -->
<!-- entities. To customize this DTD,         -->
<!-- override content values here.  -->
<!ENTITY % %Front;    "(%Title;, %Author;, %Date;, %Publisher;?, %Copyright;?)">
<!ENTITY % %Body;     "(%Part;+|%Paragraph;+)">
<!ENTITY % %Part;     "(%Number;?,%Title;?,%Paragraph;+)">
<!ENTITY % %Back;     "(%Notes;, %Appendix;*) | (%Appendix;*)">
<!ENTITY % block      "(%front;, %body;, %back;)">
<!-- Document Root                              -->
<!ELEMENT %Doctype; ( %block; )>
<!-- Document Front Matter -->
<!ELEMENT %Title; (#PCDATA)>
<!ELEMENT %Author; (#PCDATA)>
<!ELEMENT %Date; (#PCDATA)>
<!ELEMENT %Publisher; (#PCDATA)>
<!ELEMENT %Copyright; (#PCDATA)>
<!-- Document Body -->
<!ELEMENT %Part; (%part;)>
<!ELEMENT %Number; (#PCDATA)>
<!ELEMENT %Paragraph; (#PCDATA)>
<!-- Document Back Matter -->
<!ELEMENT %Notes; (%Paragraph;)+>
<!ELEMENT %Appendix; (%part;)>
```

Comentariile pot fi extinse cu niște spații albe pentru a ieși în evidență, dar acest lucru nu este făcut aici din considerentul economiei de spațiu pentru lucruri mai importante.

Apoi, asigurați-vă că numele elementelor și entităților au sens atunci când la vedeți izolate. Deși este tentant să economisiți niște acționări de taste utilizând tehno-nume stupide, cum ar fi „p" și „n" în loc de „paragraf" și „note", acestea vă pot încurca. Documentul XML ideal se auto-descrie. Nu aveți nevoie de o listă a echipei pentru a spune cine sunt jucătorii. Dacă numele folosite de dumneavoastră nu au sens pentru o persoană luată întâmplător de pe stradă, atunci este sigur că, în curând, un autor va pune cuvinte greșite într-o parte nepotrivită a documentului. Și asta nu va fi greșeala lui. Va fi a dumneavoastră. Toată lumea va ști.

De asemenea, structura fizică a DTD-ului este aranjată astfel încât toate lucrurile care se presupune că vor fi modificate sunt adunate în două locuri și sunt separate după tipul

modificării care va fi probabil efectuată. Din moment ce nici o tehnică de aranjare a codului nu este cu adevărat perfectă, acesta este cel mai bun lucru care poate fi făcut.

În continuare sunt prezentați pașii care trebuie parcurși pentru a crea un DTD blindat:

1. Ocupați-vă un timp de proiectare înainte de a scrie codul.
2. Utilizați o convenție de denumire pentru a face clară intenția de utilizare a numelor elementelor și a entităților parametru.
3. Utilizați comentarii pentru a explica detaliile semnificative. Nu pierdeți timpul notând lucruri evidente, dar dacă există un element care pare neclar, explicați oamenilor care a fost raționamentul dumneavoastră atunci când l-ați creat.
4. Fiți vorbăreți. Numele elementelor nu trebuie să arate ca și când un informatician excentric ar fi scăpat din castelul său de sticlă și ar fi ajuns la documentul dumneavoastră. Etichetați lucrurile așa cum ați dori să vedeți etichetate fișetele dacă ar trebui să găsiți datele acelui donator de sânge care vă poate salva viața.
5. Aranjați logic totul. Grupați elementele asemănătoare, iar declarațiile atributelor treceți-le imediat după elemente. Niciodată nu ar trebui să pierdeți mult timp căutând o entitate de orice fel, oriunde în DTD-ul dumneavoastră.

Cu toate că depanarea DTD-urilor poate fi dificilă, cu o proiectare bună și cu deprinderi de codare corecte, erorile posibile sunt minimizate, iar problemele sunt mai ușor de găsit decât dacă DTD-ul este un amestec de niveluri de indirectare fără legătură și fără o metodă ușoară de a identifica modul de utilizare a numelor elementelor.

Evitarea tehnicilor de codare înșelătoare

Observați că există un nivel de indirectare în numele elementelor. Acesta adaugă puțină complexitate, dar deoarece este o tehnică izolată, utilizată pentru foarte puține scopuri – stocarea și utilizarea numelor elementelor – este destul de sigură.

Totuși, puteți forța puțin această tehnică, dar în cazul unui text arbitrar nu este prea sigură. Codul următor vă arată de ce:

```
<?xml version='1.0'?>
<!DOCTYPE trickycode [
<!ELEMENT trickycode (#PCDATA) >
<!ENTITY % trickyline '&#37;trickytext;'>
<!ENTITY % trickytext '&#60;!ENTITY trickyword "flaky" >' >
%trickyline;
]>
<trickycode>This sentence demonstrates &trickyword; code.</trickycode>
```

În general, utilizarea entităților parametru pentru a construi text arbitrar este periculoasă, mai ales atunci când textul conține marcaje. Dacă greșiți, sau mai important, dacă persoana care întreține DTD-ul la ore mici, peste doi ani, greșește, poate fi extrem de dificil să găsiți o eroare având atâtea dependențe.

În linia 4 definim o linie de cod înșelătoare, cu semnul procent semnificând o entitate parametru trecută într-o secvență escape. Entitatea parametru, `%trickytext;`, nu a fost încă declarată, deci nu este extinsă atunci când semnul procent trecut într-o secvență escape este înlocuit cu caracterul procent. Este în regulă, deoarece nici nu ați utilizat încă linia. Procesorul XML înțelege că semnul procent semnifică o entitate parametru, dar nu

știe încă ce să facă cu ea. Se realizează o intrare undeva într-un tabel de simboluri, iar procesorul XML se așteaptă ca în cele din urmă totul să fie clar – și va fi.

Apoi, definim un text înșelător în linia 5. Observați că simbolul „mai mare" este trecut într-o secvență escape. Atunci când procesorul citește această linie, el realizează imediat substituția. Aici nu se mai întâmplă nimic deocamdată, singura excepție fiind că entitatea parametru `%trickyline;` are acum o valoare, `<!ENTITY trickyword "flaky" >`, care, întâmplător, este o porțiune de marcaj construită corect.

Acum, în linia 6, apelați entitatea parametru `%trickyline;`. Linia înșelătoare este extinsă, iar procesorul XML își dă seama că acum trebuie să se uite la tabelul de simboluri și să aducă definiția textului la fel de înșelător, care, întâmplător, este un mic marcaj ascuns în interiorul entității parametru `%trickyline;`. Totul este extins, iar marcajul ascuns stă acum în codul DTD-ului deghizat în entitate parametru, unde este foarte greu de văzut, și definește o entitate generală, numită `&trickyword;`, care apoi poate fi utilizată în corpul textului, la linia 8.

Deci, undeva în document, cuvântul înșelător este utilizat ca o entitate generală. El funcționează, după cum se vede și în fragmentul de cod următor, dar este într-adevăr excentric.

```
This sentence demonstrates flacky code.
```

Dacă nu ați ști dinainte, ar fi greu să vă dați seama de unde provine. Orice lucru care vă pune pe gânduri atunci când îl vedeți realizat nu este prea sigur. Poate ați uitat ce ați făcut. Poate v-ați retras în Maui. Poate un nou angajat va fi pus să modifice codul dumneavoastră elegant cu tăieturi brutale. Se poate întâmpla orice, deci cel mai bine este să vă pregătiți pentru ce e mai rău.

Asigurarea compatibilității DTD-urilor

Zonele de nume XML, prezentate amănunțit în Capitolul 6, „Zone de nume XML", ușurează foarte mult viața acelor proiectanți de DTD-uri care doresc să combine DTD-uri pentru a realiza o versiune mai mare și mai bună care să rezolve două lucruri deodată. Totuși, deocamdată ele nu sunt acceptate complet de către toți agenții utilizator. Pentru moment, singura metodă de a garanta compatibilitatea DTD-urilor este să vă asigurați că nu există nume partajate între două zone de nume, dar fără a vă baza pe disponibilitatea zonelor de nume care să rezolve conflictele de nume.

Există diverse metode de a realiza acest lucru, dar cel mai simplu este să utilizați declarațiile de entități de localizare inserate cu inteligență în codul dumneavoastră, ca o amabilitate față de vorbitorii de altă limbă decât engleza, pentru a modifica numele oricăror elemente care se află în conflict. Din acest punct de vedere, le puteți modifica pe toate cu un prefix sugestiv, economisind astfel timpul tuturor celor care, altfel, ar încerca să își dea seama de unde vine fiecare nume.

În exemplul nostru, acest lucru poate fi realizat astfel:

```
<!DOCTYPE XYZ-article
      PUBLIC "-//XYZ//Simple Article//EN"
             "article.dtd" [
<!ENTITY % Doctype       "XYZ-article">
<!ENTITY % Front         "XYZ-front">
<!ENTITY % Body          "XYZ-body">
```



```
<!ENTITY % Back          "XYZ-back">
<!ENTITY % Title         "XYZ-title">
<!ENTITY % Author        "XYZ-author">
<!ENTITY % Date          "XYZ-date">
<!ENTITY % Publisher     "XYZ-publisher">
<!ENTITY % Copyright     "XYZ-copyright">
<!ENTITY % Part          "XYZ-part">
<!ENTITY % Paragraph     "XYZ-paragraph">
<!ENTITY % Number        "XYZ-number">
<!ENTITY % Notes         "XYZ-notes">
<!ENTITY % Appendix      "XYZ-appendix">
<!ENTITY % their-dtd SYSTEM "their-dtd.dtd">
%their-dtd;
]>
```

Nu vă pasă prea mult de numele utilizate de celălalt DTD, chiar dacă și acestea sunt nume sugestive pentru paragrafe, titluri și altele, și chiar dacă un element al altui DTD se numește `article`, deoarece toate numele elementelor sunt acum prefixate cu un identificator sugestiv, ales de dumneavoastră, care poate fi modificat așa cum doriți. Acest DTD este destul de sigur chiar și fără utilizarea zonelor de nume, deoarece, în mod precaut, dumneavoastră ați ușurat viața acelor persoane care vor dori poate să modifice numele din anumite motive.

Desigur, poate că nu veți fi capabil să strângeți un alt DTD extern în această submulțime internă a DTD-ului, deci poate va trebui să mergeți la un singur DTD extern care să adune ambele DTD-uri. Trebuie numai să fiți sigur că ați redenumit elementele dumneavoastră înainte de a apela celălalt DTD.

Zonele de nume XML trebuie să le declarați și apoi să le identificați prin numele lor oriunde le utilizați. Această metodă pare mai directă și mai ușoară, deoarece puteți face modificări totale și automate în aplicația dumneavoastră printr-o comandă de editare simplă. Ea are avantajul suplimentar că nu trebuie să așteptați până când zonele de nume XML vor avea un suport complet.

Studii de caz

Construirea unui DTD pornind de la zero este o cale parcursă chiar acum de majoritatea autorilor de DTD-uri. Aceștia plănuiesc să își finanțeze cea mai mare parte a eforturilor de dezvoltare prin vânzarea accesului la DTD, deci aplicabilitatea acestor DTD-uri brevetate rămâne, cel puțin deocamdată, în domeniul business-to-business (B2B). Pentru ca publicul larg să poată utiliza un DTD, care este de cele mai multe ori locul unde stă valoarea adevărată a XML, acesta trebuie să fie disponibil în mod public.

Deși construirea unui DTD merită osteneala, din punct de vedere teoretic, numai dacă nici un DTD existent nu satisface cerințele unei anumite operații, strategia DTD-urilor brevetate a încurajat proliferarea mai multor variante pe aceeași temă. Este o dezordine greu de înțeles și care vă poate încurca, deoarece puteți compara DTD-urile brevetate concurente numai dacă le cumpărați. În cazul DTD-urilor furnizate de organizațiile din domeniu, unde membrii unui anumit consorțiu au puterea de a influența dezvoltarea de DTD-uri astfel încât să se potrivească unui scop comun, acest model poate fi satisfăcător. Dar pentru publicul larg este nevoie de un alt model.

Cu toate că există arhive de DTD-uri care au drept scop colectarea de DTD-uri pentru ca acestea să fie disponibile pentru toată lumea, aceasta ridică deseori problema locației DTD-urilor. În cazul DTD-urilor într-adevăr publice, singura soluție rațională pare să fie o locație centrală, dar, în multe situații, cea mai bună locație pentru un DTD este locația documentului care îl utilizează. Evoluția în timp a unui DTD este posibilă, dar este lipsită de înțelepciune. Dată fiind experiența noastră cu nivelul general de întreținere pe Web, pare probabil ca paginile care depind de un DTD „evoluat" să se blocheze și să treacă ani până când cineva, în afara utilizatorilor ocazionali, să observe acest lucru.

Deci, păstrarea propriilor dumneavoastră copii ale DTD-urilor „publice" poate fi o soluție rezonabilă în haosul și mișcările continue existente pe Web-ul actual. Și, desigur, păstrarea unor copii locale încurajează proliferarea de versiuni multiple ale „aceluiași" DTD, deoarece tentația de a ajusta „numai puțin" un DTD se poate dovedi prea mare pentru unii proiectanți. De asemenea, timpul de așteptare, care pe Web poate fi măsurat în ani, va împiedica multe documente să evolueze pentru a corespunde standardelor noi pe măsură ce DTD-urile publice obișnuite se modifică în timp.

Singura soluție este de a permite o diversitate infinită în perspectiva dumneavoastră asupra Web-ului. Probabil că niciodată nu veți putea să vă bazați pe o structură invariantă a Web-ului, deci orice arhitectură de documentație de pe Web trebuie să fie și ea flexibilă. Tehnologia Persistent URL (PURL), pe care o puteți vedea la `http://purl.oclc.org/`, vă poate ajuta să oferiți o anumită continuitate în unele situații. Dar, probabil că va fi necesar un echivalent Web al Bibliotecii Congresului SUA sau al bibliotecilor universităților importante, care va arhiva copii ale documentelor și DTD-urile asociate acestora, referite în mod obișnuit de alții, pentru a satisface cerințele referințelor științifice stabile. Proiectul Gutenburg (`http://www.promo.com/`) și Bibliomania (`http://www.bibliomania.com/`) sunt primele încercări (non-XML) de a satisface această nevoie, dar există nenumărate altele cu țeluri similare, printre care multe bazate cu fermitate pe SGML sau XML, cum ar fi Text Encoding Initiative (TEI), la `http://www.tei.uic.edu/orgs/tei/`, sau Women Writers Project de la Universitatea Brown (`http://www.wwp.brown.edu/`).

Aceste ultime proiecte militează pentru arhivarea atât a documentelor XML cât și a DTD-urilor lor, astfel încât studenții (sau alții) care doresc să utilizeze texte clasice să aibă un depozit central (sau măcar stabil) unde să le poată găsi. O bibliotecă stabilă face posibilă încorporarea de documente XML și DTD-uri legate sau transcluse fără a risca pierderea continuității care ar rezulta din pierderea locației referinței.

Zone de nume XML

CAPITOLUL 6

În acest capitol

- Utilizarea mai multor DTD-uri cu zonele de nume
- Declararea unei zone de nume
- Utilizarea numelor calificate
- Identificarea zonelor de nume în elemente și în atribute
- Lucrul fără zone de nume
- Declararea zonelor de nume în limbi non-europene
- Utilizarea zonelor de nume în alte contexte
- Utilizarea zonelor de nume în moduri non-standard
- Studii de caz

Utilizarea mai multor DTD-uri cu zonele de nume

Zonele de nume XML furnizează o metodă de reutilizare a marcajelor în documente, chiar dacă există conflicte între numele elementelor și atributelor folosite în oricare dintre marcaje. Un conflict înseamnă că un element (sau o combinație element/atribut) definit într-un DTD are același nume cu un element definit într-un alt DTD. Prin referirea explicită a zonei de nume în care este definit un anumit marcaj, instanțe multiple de zone de nume externe pot fi utilizate și luate în considerare într-un singur document. Zonele de nume există cu scopul de a face unice numele de elemente și de atribute. Specificația care le definește nu menționează un alt motiv pentru existența lor. După cum veți vedea în secțiunea „Utilizarea zonelor de nume în moduri non-standard" din acest capitol, aceasta nu a împiedicat utilizarea lor în multe alte contexte.

Notă O *zonă de nume* înseamnă pur și simplu contextul în care un anumit nume este utilizat, având niște garanții referitoare la unicitatea sa. În XML, o zonă de nume poate fi legată de DTD-ul utilizat pentru a defini un set de etichete și atribute prin utilizarea URL-ului său ca identificator unic, deoarece un DTD garantează unicitatea numelor. Dar orice identificator unic poate fi folosit pentru a defini o zonă de nume, nu numai URL-ul unui DTD. În alte contexte, cum ar fi utilizarea unui număr ISBN ca identificator al unei zone de nume, unicitatea poate să nu fie garantată, deoarece autoritatea de denumire, care alocă numere ISBN unice, nu definește un DTD asociat cu acestea.

Dacă în același document sunt utilizate mai multe DTD-uri, ceea ce se poate întâmpla cu ușurință atunci când încercați să utilizați DTD-uri din alte medii, procesorul XML nu are cum să decidă *a priori* dacă o anumită etichetă `<table>` vine din zona de nume XHTML și nu din zona de nume MathML. Deseori domnește confuzia. Cu toate că zonele de nume XML sunt o recomandare a W3C, ele nu sunt încă universale, dar puteți conta pe implementarea lor, în scurt timp, în majoritatea contextelor, deoarece sunt destul de valoroase.

Ca un exemplu, cercetați chiar zona de nume a XML:

```
xmlns:xml="http://www.w3.org/XML/1998/namespace"
```

Deoarece W3C are control complet asupra propriei zone de nume, zona de nume poate fi, și este, garantată de W3C ca fiind unică. Orice specificație XML viitoare poate fi atribuită cu ușurință unei alte zone de nume pentru a marca diferența dintre ele. Totuși, observați că zonei de nume XML i-a fost atribuit prefixul `xml:` de către codul precedent. Aceasta este valoarea ei prestabilită pentru, și numai pentru, XML. Nici unei alte zone de nume nu i se poate atribui acest prefix, iar încercarea de a face acest lucru va fi o eroare, indiferent de combinația de tipuri de litere. Deci este neclar cum poate fi prefixul `xml:` asociat unei noi versiuni a XML și ce anume va face cu el un agent utilizator care suportă zonele de nume. Sintaxa corectă necesară pentru declararea și utilizarea prefixelor zonelor de nume este cercetată în secțiunea următoare, „Declararea unei zone de nume".

De asemenea, trebuie să știți că există un RFC-2611 care se referă la URN Namespace Definition Mechanisms (mecanismele URN (Uniform Resource Name – schemă de denumire a resurselor) de definire a zonelor de nume). Acesta nu este legat în mod direct de zonele de nume XML, cu toate că specificația XML Namespace face referiri la RFC 2141, *URN Syntax*, care, la rândul său, se referă la aceasta. Zonele de nume în aceste alte contexte sunt dezbătute în secțiunea „Utilizarea zonelor de nume în alte contexte" din acest capitol. Un URN este un mecanism alternativ pentru identificarea unei zone de nume, care va fi discutat în acea secțiune.

Declararea unei zone de nume

Zonele de nume sunt declarate utilizând un atribut XML universal, numit `xmlns`. Acest atribut este disponibil în fiecare element XML chiar dacă nu a fost declarat într-un DTD și este recunoscut de un analizor care validează chiar și fără această declarație. Este o parte a limbajului XML în sine.

Creați o zonă de nume prin includerea unui atribut `xmlns` în eticheta de pornire sau în elementul rădăcină al documentului sau al fragmentului. Domeniul de valabilitate al zonei de nume este format din acel element și toți descendenții săi. Ca un exemplu, studiați un document XHTML simplu. XHTML este interesant în mod deosebit deoarece a fost proiectat astfel încât să poată fi extins cu ușurință utilizând alte DTD-uri XML și pentru că necesită ca o zonă de nume să fie declarată în fiecare document XHTML pentru ca acesta să fie considerat valid:

1. Zona de nume pentru XHTML 1.0 este `http://www.w3.org/1999/xhtml`, deci elementul rădăcină `<html>` trebuie să declare cel puțin această zonă de nume:

```
…
<html xmlns="http://www.w3.org/1999/xhtml">
…
</html>
```


Dacă nu sunt utilizate alte zone de nume sau dacă este zona de nume principală, ea poate fi declarată ca zonă de nume prestabilită, fără a fi nevoie de prefix.

2. Și utilizarea DTD-ului unui alt document în XHTML este destul de simplă. În acest caz, cel puțin o zonă de nume trebuie să fie apelată explicit printr-un prefix, cum ar fi de exemplu:

```
…
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:doc="http://www.oasis-open.org/docbook/">
…
</html>
```

Notă DocBook nu are declarată o zonă de nume oficială, deci declarați dumneavoastră una. Prefixul declarat pentru DTD-ul DocBook este „doc" și, de fapt, este adăugat ca *sufix* la declarația `xmlns` utilizând o sintaxă specială, asociată numai cu atributul `xmlns`:

```
xmlns:doc="…"
```

3. După ce ați declarat zona de nume DocBook, puteți să utilizați etichetele DocBook cu prefixul declarat în documentul dumneavoastră XHTML, astfel:

```
…
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:doc="http://www.oasis-open.org/docbook/">
  <head>
    <title>DocumentTitle</title>
  </head>
  <body>
    …
    <doc:title>BookTitle</doc:title>
    …
  </body>
</html>
```

Alternativ, puteți utiliza etichete DocBook fără a le declara în prealabil, incluzând zona de nume DocBook în fiecare etichetă DocBook, astfel:

```
…
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>DocumentTitle</title>
  </head>
  <body>
    …
    <title xmlns:doc="http://www.oasis-open.org/docbook/">BookTitle</title>
    …
  </body>
</html>
```

Acest lucru are același efect cu declararea unei zone de nume prefixate și apoi utilizarea prefixului, deoarece, prin definiție, prefixul zonei de nume este doar o prescurtare pentru zona de nume efectivă. Pe plan intern, agentul utilizator care suportă

zonele de nume trebuie să utilizeze zona de nume efectivă, și nu prefixul, pentru a garanta unicitatea. Dar utilizarea zonelor de nume cu domeniu de valabilitate mai mic decât cel global poate ridica probleme pentru CSS și alte mecanisme care nu înțeleg perfect zonele de nume. Majoritatea implementărilor actuale înțeleg numai sintaxa cu prefix, deci, în practică, zonele de nume la nivel de atribut nu sunt echivalente cu zonele de nume globale, chiar dacă din punct de vedere teoretic așa ar trebui.

4. După ce o zonă de nume prefixată este declarată, puteți adresa atribute din acea zonă de nume utilizând numele atributului cu prefixul zonei de nume, dar acest lucru poate duce la interacțiuni complexe, confuze și, în ultimă instanță, nedefinite, între agentul utilizator și DTD, dacă nu aveți grijă. De exemplu, deoarece zonele de nume XML sunt sensibile la tipul literelor, în timp ce URI-urile de pe Web nu sunt, codul următor este corect în contextul zonelor de nume XML, dar nu și în contextul XML sau XHTML:

```
…
<html xmlns:xhtml="http://www.w3.org/1999/xhtml"
      xmlns:xht="HTTP://WWW.W3.ORG/1999/XHTML">
  …
  <body xhtml:title="FirstTitle" xht:title="SecondTitle">
    …
  </body>
</html>
```

Zonele de nume XML nu recunosc faptul că cele două zone de nume din exemplul precedent sunt echivalente, singura diferență dintre ele fiind tipul literelor. Deci ilegala atribuire dublă de valori la același atribut XML trece neobservată.

La începutul fiecărui document XHTML există o declarație obligatorie a atributului `xmlns`, care îl declară ca fiind parte a zonei de nume XHTML, ca și cea care urmează:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Această declarație stabilește, de asemenea, zona de nume prestabilită, deoarece zonei de nume nu îi este asociat un prefix de zonă de nume. Prefixul zonei de nume este o parte a sintaxei extinse a atributului `xmlns` care vă permite să setați un prefix mnemonic asociat unei zone de nume, care poate fi utilizat în nume de elemente pentru ca dumneavoastră să puteți urmări cu ușurință ce element utilizați în orice moment. Ca un exemplu, codul următor declară zona de nume pentru HTML 4.0 și îi asociază un prefix pentru a putea fi utilizat în combinație cu zona de nume XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:htm="http://www.w3.org/TR/REC-html40">
```

Fragmentul precedent declară ca prestabilită zona de nume XHTML, deci orice nume fără un prefix se presupune că face parte din această zonă de nume. Apoi, declară un prefix, `htm`, utilizând o sintaxă specială, separată prin două puncte, pentru atributul `xmlns` și îi asociază un URI, zona de nume oficială a HTML 4.0.

Sugestie de la LeeAnne.com Words to weave by

Utilizarea unei zone de nume prestabilite vă permite să săriți peste adăugarea acelor prefixe urâte la fiecare etichetă. Puteți să vă gândiți care sunt etichetele pe care le utilizați cel mai des și să le setați pe acelea ca fiind prestabilite, aceasta permițându-vă să scăpați de mult tastat și făcând codul dumneavoastră mai ușor de citit. De asemenea, poate să vă permită să reutilizați aproape oricare foaie de stil pe care ați utilizat-o la paginile dumneavoastră, adăugând proprietățile și valorile foii de stil numai la etichetele noi.

Nu numai că asociați prefixul unei zone de nume cu zona de nume atunci când o deschideți, dar sunteți capabil să utilizați acel prefix pentru a eticheta chiar eticheta `<html>`. Întâmplător, acesta este elementul rădăcină pentru acest arbore de document, deci setați domeniul de valabilitate al zonei de nume prestabilite astfel încât să acopere întregul document. Cele două atribute `xmlns` nu sunt de fapt identice, ceea ce ar fi ilegal atât în HTML cât și în XML, deoarece ele sunt în zone de nume diferite, după cum este indicat prin adăugarea ca sufix la cel de-al doilea atribut `xmlns` a unui prefix de zonă de nume, chiar dacă ele împart aceeași etichetă. Totul devine ușor confuz.

Sugestie de la LeeAnne.com Words to weave by

În exemplele precedente, în ciuda faptului că ambele zone de nume sunt URL-uri, o submulțime a URI-urilor, la nici una dintre locații nu se găsește în prezent nimic care să poată fi citit pentru a descoperi ce este zona de nume. O zonă de nume o puteți numi Fred (un URL relativ și, prin urmare, un URI) atât timp cât sunteți consecvent. Prefixul este doar un înlocuitor pentru numele complet al zonei de nume și poate fi înlocuit oricând prin acesta utilizând atributul `xmlns`.

Acum, există două zone de nume care sunt aproape identice. Fiecare are o etichetă `<p>`, o etichetă `<h1>` și așa mai departe. Dar le puteți deosebi printr-un mecanism de adresare simplu:

Eticheta `<p>` din zona de nume XHTML este pur și simplu eticheta `<p>`. În mod prestabilit, toate etichetele necalificate fac parte din această zonă de nume. Eticheta `<p>` din zona de nume HTML 4 poate fi adresată ca

```
<htm:p>This is some text in the HTML 4 namespace.</htm:p>
```

Același principiu se aplică și atributelor.

Puteți chiar să accesați atribute din ambele zone de nume în interiorul aceleiași etichete. Iată cum:

```
<img src="image.gif"    htm:src="picture.gif"
     alt="[Image]"      htm:alt="[Image]" >
```

Ce va face cu această declarație ciudată un browser care are suport pentru zonele de nume rămâne ca exercițiu pentru imaginația bogată a cititorului și pentru capriciile dezvoltatorilor de browsere.

Aceasta este, pe scurt, întreaga poveste a zonelor de nume XML. Utilizarea zonelor de nume permite pur și simplu unui agent utilizator sau unui browser să facă distincție între etichetele și atributele altfel ambigue. Deoarece zonele de nume sunt atât de importante și deoarece utilizările non-standard ale zonelor de nume au devenit uzuale, le veți studia mai amănunțit în secțiunile următoare, dar aproape orice ați dori să faceți este prezentat în această secțiune.

Utilizarea numelor calificate

Un *nume calificat* este pur și simplu partea locală a numelui, fie acesta elementul în sine sau unul dintre atributele sale, plus prefixul care servește ca înlocuitor pentru numele complet al zonei de nume. De fiecare dată este posibil să utilizați numele complet al zonei de nume în loc de, sau pe lângă prefix, iar acest lucru ar trebui făcut pentru elementele al căror domeniu de valabilitate se întinde dincolo de documentul curent.

Deci etichetele <p> următoare sunt echivalente, presupunând că am declarat prefixul de zonă de nume htm în modul prezentat anterior:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:htm="http://www.w3.org/TR/REC-html140">
<htm:p>Some text in the HTML 4.0 namespace.</htm:p>
<p xmlns="http://www.w3.org/TR/REC-html140"> Some text in the HTML 4.0
namespace.</htm:p>
```

Puteți considera prefixul ca fiind o metodă prescurtată de introducere a întregului atribut xmlns în fiecare etichetă.

Deoarece atributele nu pot fi asociate altor atribute, utilizarea unui prefix de zonă de nume este singura metodă de a asocia o zonă de nume cu un atribut.

Identificarea zonelor de nume în elemente și în atribute

Aici avem un exemplu de fragment de pagină Web care utilizează două zone de nume, o versiune experimentală a recenziei unei cărți, făcută de mine, la http://www.cybergrrl.com/review/ și specificația XHTML în sine:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:wbo="http://www.leeanne.org/bookreview">
  [...]
<wbo:entry>
   <dt>
      <img src="wbfiction.gif" alt="Fiction" height="11"
           width="25" align="BOTTOM" />
      <wbo:link><a href="q1-2_1999.html#Saturday">
         <wbo:name>Dodici Azpadu's</wbo:name>
         <wbo:title><cite>Saturday Night in the Prime of Life</cite></wbo:title></a>
      </wbo:link><br />
   </dt>
      <dd>
         Reviewed by
         <wbo:reviewer>Lee Anne Phillips</wbo:reviewer>
         <wbo:issue>(Q1-2, 1999)</wbo:issue>
         <wbo:category>(F)</wbo:category>
      </dd>
</wbo:entry>
```

Zona de nume XHTML este cea prestabilită, deci pentru etichetele XHTML 1.0, care dau documentului o structură de afișare fără ambiguități, nu aveți nevoie de nume calificate. Zona de nume a paginilor de recenzie formează o structură de date XML proiectată special pentru indexul recenziilor. Prin intercalarea etichetelor XML cu etichetele XHTML obțineți o pagină care va fi afișată corect în aproape orice browser și permite în continuare accesul ușor la câmpurile de date.

Figura 6.1 prezintă codul paginii de recenzie a cărții într-un mediu de editare care nu are suport pentru XML, care poate fi de fapt instrumentul de editare obișnuit pentru fragmentele mici de XML din fișiere mai mari, în cazul în care browserele cu suport XML nu vor reuși să își impună prezența universală pe calculatoare. Figura 6.2 prezintă același

document vizualizat într-un browser care nu are deloc suport pentru XML. După cum puteți vedea, etichetele înglobate din zona de nume XHTML permit browserului moștenit să se comporte rezonabil. Dat fiind faptul că majoritatea utilizatorilor își actualizează browserele numai la achiziționarea unui calculator nou, probabil că vor trece mulți ani până când browserele cu suport pentru XML se vor apropia de universalitate. De asemenea, este puțin probabil ca mediile de editare XML să existe pe fiecare calculator, deci multe pagini combinate HTML/XML vor fi create utilizând instrumentele cele mai familiare utilizatorului.

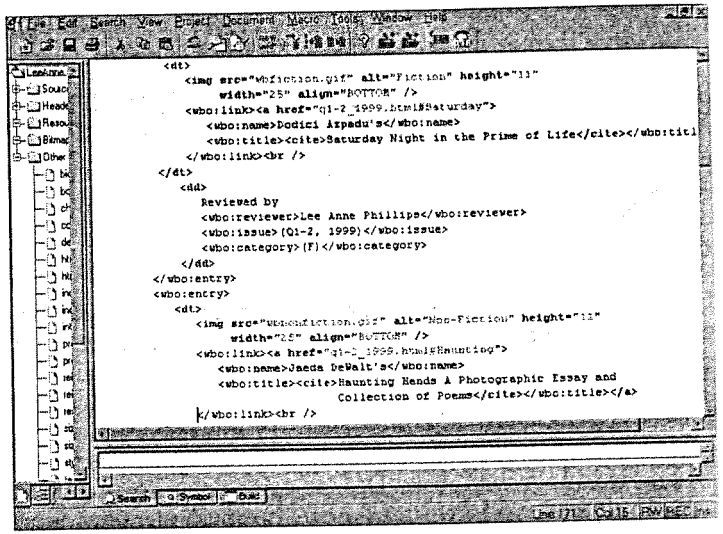


FIGURA 6.1
Puteți utiliza editorul pentru programatori Visual SlickEdit pentru a vizualiza structura unei pagini Web XML care utilizează zone de nume XML pentru a intercala etichetele.

Aceasta este o utilizare obișnuită a zonelor de nume, folosind HTML sau XHTML pentru a „înfrumuseța" sau a „aranja frumos" pagina, ținând cont de faptul că majoritatea lumii nu poate să facă nici un fel de foi de stil pentru etichetele XML.

Editarea XML

Cu toate că există multe instrumente specializate pentru editarea fișierelor XML, în majoritatea situațiilor acest autor se simte mult mai bine folosind un editor adevărat pentru programatori. Visual SlickEdit emulează Vi, Emacs și mult regretatul Brief, toate fiind editoare de lux pentru excentricii calculatoarelor. Cu toate că este foarte interesant să avem meniuri autoderulante pentru atribute, expresiile regulate sunt mai frumoase. În plus, puteți să scrieți propriile dumneavoastră macrocomenzi dacă nu vă place modul în care funcționează.

Preocuparea pentru accesibilitate devine mai importantă pe măsură ce Web-ul devine saturat de afișarea grafică a oricărui lucru. Utilizând zone de nume și CSS sau XSL, puteți afișa conținut text în mod selectiv, într-o formă accesibilă, în timp ce imaginile strălucitoare le faceți disponibile pentru cei care le doresc și pot să le utilizeze.

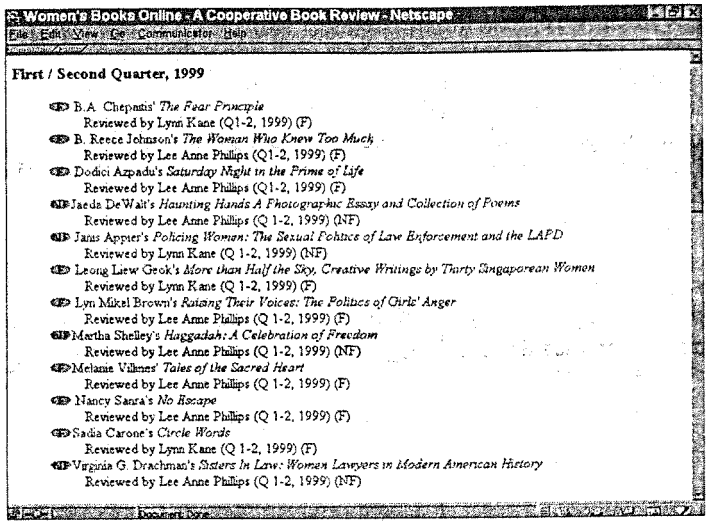


FIGURA 6.2

Aceeaşi pagină, afişată în Netscape Navigator 4.7. Etichetele XML nu au nici un efect pe ecran în acest model, deoarece este destinat browserelor accesibile, iar această ediţie de Navigator nu cunoaşte nimic despre XML.

Deoarece zonele de nume XML sunt de fapt doar nume arbitrare, puteţi eticheta aproape orice cu ele pentru alte scopuri decât partajarea câtorva etichete. În aranjarea unei afaceri, puteţi eticheta categorii diferite de mărfuri cu propriile lor zone de nume, dar utilizând aceeaşi structură de înregistrări a bazei de date. Aceasta vă permite, de exemplu, să trataţi diferit pentru afişare ciocanele (hammers) de hamacuri (hammocks) şi de plante decorative (hemlocks), chiar dacă ele sunt doar nişte intrări identice într-un sistem de inventariere şi aprovizionare. De fapt, orice situaţie în care doriţi să faceţi distincţie între instanţe ale aceleiaşi înregistrări poate fi supusă identificării prin zone de nume.

Acele zone de nume diferite le puteţi defini astfel:

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:hammer="http://www.leeanne.org/hammers"
      xmlns:hammock="http://www.leeanne.org/hammocks"
      xmlns:hemlock="http://www.leeanne.org/hemlocks">
```

Apoi, puteţi face aşa ceva:

```
<hammer:price>$29.95</hammer:price>
<hammock:price>$29.95</hammock:price>
<hemlock:price>$29.95</hemlock:price>
```

şi să utilizaţi o foaie de stil pentru a reda uneltele cu negru, mobilierul de grădină cu verde, iar articolele de horticultură cu maro. Sau puteţi seta o transformare XSL pentru a genera trei cataloage diferite pentru a lista toate ciocanele într-un loc, toate hamacurile în altul şi toate plantele decorative într-un al treilea.

Reciproc, puteţi eticheta instanţe multiple ale unor structuri de înregistrări total diferite, pentru a permite aplicaţiilor dumneavoastră să le deosebească pentru a evita conflictele în zonele de nume. Pentru acest lucru au fost proiectate zonele de lucru şi ele sunt comode, după cum s-a văzut în exemplele HTML 4.0 şi XHTML 1.0 anterioare.

Lucrul fără zone de nume

În Capitolul 4, „Extinderea definiţiei tipului documentului prin modificări locale", aţi văzut posibilitatea simplei suprascrieri a numelor elementelor în locul utilizării zonelor de nume, atunci când încercaţi să utilizaţi DTD-uri modulare. Deoarece zonele de nume XML nu se adresează de fapt problemei DTD-urilor modulare, pentru că nu puteţi fuziona DTD-urile în sine ci doar să urmăriţi de unde vine o anumită etichetă, acesta poate fi un mecanism mult mai potrivit în aşteptarea schemelor XML.

Suprascrierea numelor are şi un efect secundar şi anume acela că crearea de documente utilizând nume de elemente suprascrise va face ca aceste nume să nu poată fi recunoscute de nici o foaie de stil care a fost setată pentru zona de nume a documentului original. De asemenea, creează etichetele într-o formă care le face să nu poată fi recunoscute fără a accesa ambele submulţimi ale DTD-ului, atât cea internă cât şi cea externă. Dar utilizarea zonelor de nume XML este aproape la fel de dificilă. Trebuie fie să adăugaţi nume calificate la foaia dumneavoastră de stil pentru a le putea utiliza, fie să aşteptaţi un viitor sistem de aliasuri pentru zonele de nume care să permită foilor de stil să refere zonele de nume efective şi nu prefixele. Aceasta este o problemă atât de obişnuită, deoarece autorii de documente îşi culeg propriile lor prefixe în timp ce zona de nume este, din punct de vedere teoretic, atribuită, încât grupul de lucru la zonele de nume XML caută tocmai un astfel de sistem automat de armonizare a prefixelor zonelor de nume.

Aceeaşi problemă există şi la suprascrierea numelor într-o limbă străină (alta decât engleza). Claritatea obţinută pentru autorii vorbitori de altă limbă decât cea engleză poate să compenseze incomoditatea utilizării unui instrument de transformare pentru a converti documentele terminate înapoi la terminologia DTD standard, pentru publicarea mondială. Cu toate că în XSL puteţi teoretic să rescrieţi rapid descrierile etichetelor, experienţa a arătat că uneori este necesar să studiaţi amănunţit coduri XML sau chiar SGML, fără a beneficia de medii de editare moderne.

Zonele de nume reprezintă doar o soluţie stabilă care permite ca o muncă utilă să fie terminată înainte de apariţia schemelor XML. Cu toate că zonele de nume vă permit să utilizaţi nume de elemente din zone de nume diferite, zonele de nume XML nu spun nimic despre ce înseamnă numele sau ce tip de dată conţin. Ele nu sunt un înlocuitor al formelor arhitecturale SGML sau a orice altceva.

Utilizarea zonelor de nume într-un document este echivalentă cu crearea unui nou tip de document. Într-o oarecare măsură se modifică aproape totul. Vă veţi aştepta să aveţi de creat o foaie de stil care să o însoţească şi să petreceţi ceva timp pentru a face totul corect. Cu toate că puteţi salva o cantitate considerabilă de muncă prin alegerea corectă a zonei de nume prestabilite, astfel încât foaia dumneavoastră de stil să rămână în mare măsură aceeaşi, totuşi veţi avea ceva de lucru. Dar cel mai bun lucru în privinţa zonelor de nume este că ele funcţionează, chiar dacă numai în domeniul lor limitat, şi vă permit să lucraţi până când toate se vor clarifica.

Declararea zonelor de nume în limbi non-europene

Definiţia actuală a zonelor de nume are probleme serioase de internaţionalizare, deoarece restricţia lor la sintaxa URI aduce dependenţa de US-ASCII, care din motive

istorice este *limba comună* a Internet-ului. Cu toate că este foarte comod pentru unii utilizatori – de exemplu, cititorii acestei cărți în limba engleză – o altă categorie de utilizatori este dezavantajată semnificativ – adică orice vorbitor de limbă care nu se bazează pe alfabetul latin. Majoritatea limbilor europene actuale utilizează caractere sau diagrafe formate dintr-un caracter de bază și un accent sau alt simbol distinctiv care extind alfabetul latin pentru a acoperi cerințele limbii respective. Deoarece US-ASCII este atât de omniprezent, au fost dezvoltate standarde precise pentru transformarea mai mult sau mai puțin automată a cuvintelor efective într-o formă echivalentă care poate fi reprezentată în US-ASCII, cum ar fi germanul ü (u-umlaut), în Bücher, care poate fi scris ca „ue".

Dar unele tastaturi non-engleze pot să nu conțină caractere US-ASCII deoarece nu sunt utilizate în limba vorbită de utilizatorul tastaturii. O tastatură rusească, de exemplu, poate să nu aibă nici un caracter latin, cu toate că poate exista o asemănare de suprafață între anumite caractere chirilice și litere latine. În același mod, pe o tastatură arabă găsiți caractere care nu seamănă deloc cu caracterele latine, deci un sit numit `www.zebra.com` poate fi extrem de greu de introdus de cineva care utilizează astfel de tastaturi. Dacă mai adăugați și faptul că standardul URI existent acceptă numai text scris de la stânga la dreapta, în timp ce araba și multe alte limbi utilizează scrisul de la dreapta la stânga, dificultatea sporește.

Și indiferent de comoditate, un URI utilizând un alfabet latin nu este mnemonic deloc pentru cei care nu sunt obișnuiți cu un astfel de alfabet. Iar mnemonicile contează. Companiile cheltuiesc milioane de dolari pentru domenii Internet care să fie ușor de reținut și de scris. În ciuda faptului că Web-ul a fost dominat timp de mai mulți ani de limba engleză, sau în mare de limbile europene, valul de interes pentru Internet din partea Chinei, Japoniei și a altor puteri economice neoccidentale presupune că în cele din urmă vor trebui realizate facilități și pentru acestea.

Au fost făcute propuneri pentru extinderea URI-urilor astfel încât să accepte setul de caractere UTF-8 ISO-10646, dar toate acestea au și dezavantaje importante, printre care transmiterea de bâlbâieli intermediare cum ar fi `http://www.أليس.om/` care sunt greu de înțeles și utilizat dacă suportul nativ pentru aceste tipuri de decodificări nu este încorporat în agentul utilizator. Ar trebui menționat că probleme asemănătoare există și pentru codificările de caractere EBCDIC utilizate în lumea mainframe-urilor IBM.

Pe termen scurt, pare destul de probabil că nu se va face nimic pentru ca domeniile Internet să fie mai comode și mai mnemonice pentru utilizatorii internaționali, deoarece o foarte mare parte a infrastructurii rețelei ar trebui actualizată. Dar presiunile exercitate pentru modificare cresc pe măsură ce națiunile în curs de dezvoltare devin centre de forță economică. India, de exemplu, este raportată ca având lățime de bandă pe Internet disponibilă pe cap de locuitor mai mare decât cea a Statelor Unite, iar companiile înființate în India cresc ca putere și intră pe piață. Cu toate că India utilizează în parte limba engleză există, de asemenea, și multe alte scrieri regionale sau naționale utilizate zilnic.

Utilizarea zonelor de nume în alte contexte

Zonele de nume sunt utilizate și în alte contexte decât XML, deoarece identificatorii unici au multe domenii de utilizare în informatică, dar identificatorii unici sunt, de asemenea, facilități valoroase și pot depinde de restricții de brevetare sau, atunci când se referă la oameni, de preocupări de confidențialitate și legalitate. În aceste contexte, o

zonă de nume este identificată de obicei printr-un URN (Uniform Resource Name). Totuși, același mecanism URN poate identifica o zonă de nume în XML, deci trebuie să fiți atent la limitările identificatorilor URN.

RFC-2611, *mecanismul definirii zonelor de nume URN* (*URN Namespace Definition Mechanism*), se supune unei alte metode de definire a zonelor de nume, bazată pe recunoașterea autorităților centrale responsabile cu administrarea identificatorilor URN. RFC-2169, *o convenție simplă pentru utilizarea HTTP în rezoluția URN* (*A Trivial Convention for Using HTTP in URN Resolution*), schițează un mecanism ipotetic de rezoluție URN. Dar în practică, este probabil ca nici una dintre aceste metode și protocoale să nu facă mare lucru. Autoritățile emitente și persoanele care răspund de această informație este improbabil să o ofere gratuit, iar utilizatorii ocazionali ai informației este puțin probabil să plătească pentru a rezolva sau verifica referințe URN aleatoare.

Numerele seriale internaționale standard (ISSN – International Standard Serial Number), de exemplu, sunt administrate de Centrul Internațional de ISSN din Paris, `http://www.issn.org/`. În realitate, el are o bază de date în care se poate căuta (contra cost) pentru a identifica un număr serial bazat pe un ISSN, lucru rar între „autorități". Includerea numerelor seriale, în acest context, se referă la reviste și alte publicații periodice.

Spre deosebire de acestea, numerele ISBN sunt atribuite de un număr uimitor de agenții și subagenții, inclusiv edituri, astfel încât identificarea unui anumit articol poate să nu fie, de fapt, posibilă dacă editura nu mai activează și nu puteți găsi referiri la el în nici un alt loc. Există o autoritate mai mult sau mai puțin centrală întreținută de editura R.R. Bowker, dar până și această autoritate este împărțită între Statele Unite (Bowker) și agențiile individuale din fiecare țară, țările care au legături istorice și lingvistice partajând o agenție comună.

Deoarece Bowker se ocupă de publicarea de cărți, inclusiv cataloage de apariții conținând chei pentru descoperirea numerelor care au fost atribuite deja unor submulțimi mari de cărți publicate, găsirea unui număr ISBN arbitrar este posibilă utilizând o referință pe hârtie. Și deoarece Bowker mai publică și ghiduri pentru găsirea prefixelor atribuite anumitor edituri, este posibil din punct de vedere teoretic să găsiți orice număr ISBN. Dar nu există nici o metodă on-line pentru găsirea tuturor referințelor și probabil nu va exista niciodată. Unele subagenții și edituri este posibil să nu aibă nici măcar adresă de e-mail, cu atât mai puțin situri Web, deci singurul mecanism universal de identificare este foarte lent. În ultimă instanță trebuie să ne bazăm pentru acest scop pe agențiile poștale fizice ale fiecărei țări, fapt care nu este o soluție satisfăcătoare pentru majoritatea cererilor în timp real, în ciuda faptului că formatul unei zone de nume bazate pe ISBN pare să implice un fel de oportunitate on-line: `xmlns="URN:ISBN:0789719967"`. Informații despre ISBN-uri pot fi găsite la `http://www.isbn.org/`.

În mod asemănător, un număr de asigurare socială din S.U.A. îndeplinește anumite cerințe ale unei zone de nume; este unic din punct de vedere teoretic și este atribuit de către o autoritate centrală. Dar din motive de confidențialitate, un număr de asigurare socială nu poate să corespundă cu o identitate pentru majoritatea solicitanților și nici nu este ușor de verificat. Iar escrocheriile privind asigurările sociale sunt atât de frecvente încât este posibil ca unele numere să fie utilizate de sute de persoane, sau persoana efectivă referită „oficial" prin acel număr este decedată iar cei de la asigurările sociale nu știu

acest lucru, sau poate că nici o persoană nu a fost referită de un anumit număr complet fals. Dar aceasta nu înseamnă că nu există înregistrări conținând aceste numere. Frauda prin asigurările sociale se bazează tocmai pe introducerea acestor numere în baze de date, deci le puteți găsi peste tot.

Întreaga idee de zonă de nume din acest punct de vedere este puțin neliniștitoare deoarece presupune baze de date universal accesibile conținând informații detaliate disponibile pentru oricine le dorește. Când este aplicată pentru oameni, problema este evidentă, dar chiar și când este aplicată obiectelor fizice există probleme filozofice și practice.

Este discutabil faptul că *trebuie* să existe un cod de produs într-adevăr universal pentru cărți. Aceasta ar însemna că trebuie să plătiți la o anumită autoritate, cum ar fi `ISBN.org`, pentru a avea privilegiul de a publica o carte. În multe cazuri poate fi imposibilă sau nu este deloc recomandabilă înregistrarea unei cărți din moment ce autoritatea responsabilă poate fi un regim totalitar sau represiv care dorește interzicerea anumitor subiecte. Și nici chiar acest lucru nu ia în considerare operele de artă sau alte lucrări unice. Dacă cerem ca Mona Lisa să aibă un număr UPC (Uniform Product Code – sistem de identificare a produselor comerciale cu ajutorul codurilor de bare) și să aibă pe verso un cod de bare, atunci poate că va trebui să regândim întregul nostru concept de artă, și poate chiar îl vom distruge prin acest proces. Dacă totul trebuie definit anticipat, sau chiar doar definit, atunci libertatea cuvântului și arta sunt în pericol.

Poate fi chiar un paradox existențial. Este posibil chiar să nu existe lucruri ca identificatori unici și universali atunci când se aplică persoanelor efective sau obiectelor. O persoană având cetățenie dublă și două pașapoarte nu reprezintă două persoane diferite numai pentru că două jurisdicții diferite au atribuit acelei persoane un identificator unic diferit. Dar când aceeași persoană are două pașapoarte obținute în mod fraudulos de la aceeași jurisdicție, este o altă poveste. Scopul ultimei fraude putem presupune că este acela de a *pretinde* că sunt două persoane, iar succesul fraudei presupune că oricine având importanță oficială, va crede că *există* două persoane diferite. Deci în acest sens restrâns, pretenția se apropie de realitate. Iar o persoană care a reușit să nu fie identificată de nici un stat sau agenție este de presupus că este totuși reală, chiar dacă nu există date oficiale care să spună acest lucru.

Deci, dacă tot ce putem face este numai să *aproximăm* această structură ideală de identificare unică, este probabil nejustificat să ne comportăm ca și când am putea, de fapt, să ne bazăm pe o perfecțiune care nu există. Aproape toți am trăit experiența încercărilor zadarnice ale birocraților de a-și redobândi calmul după ce o încălcare a regulamentului i-a scos din sărite și i-a făcut să refuze cu îndârjire să recunoască realitatea; ar trebui să creăm sisteme care să nu fie chiar atât de fragile.

Utilizarea zonelor de nume în moduri non-standard

După ce zonele de nume XML ale W3C au definit un atribut universal garantat ca fiind disponibil în fiecare DTD, câțiva programatori cu inițiativă s-au hotărât să supraîncarce zonele de nume cu alte contexte. Cel mai faimos exemplu este Microsoft care a utilizat, de altfel, non-standardul `xmlns:HTML="http://www.w3.org/Profiles/XHTML-transitional"` ca o zonă de nume specială și prefix de zonă de nume rezervat, transformându-l într-un

comutator care să activeze prelucrarea foii de stil speciale pentru elemente XHTML din Internet Explorer 5. De asemenea, `<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">` este utilizat ca un comutator pentru a activa prelucrarea foii de stil XSL în Internet Explorer.

Dar utilizări asemănătoare există din belșug inclusiv în cadrul W3C. SMIL, de exemplu, are de gând în cele din urmă să utilizeze o zonă de nume XML pentru a identifica extensiile la SMIL folosind un atribut `system-required` a cărui valoare este o zonă de nume. Cu alte cuvinte, zona de nume este utilizată ca un comutator pentru a identifica dacă o anumită implementare a SMIL acceptă sau nu o extensie indicată de acea zonă de nume.

Aceste utilizări depășesc cu mult nevoia simplă de identificare clară a etichetelor și a atributelor lor și extind zonele de nume într-un rol metalingvistic de arbitru al sintaxei și structură de control simplă, în loc să îl păstreze ca un mecanism pentru transformarea numelor astfel încât acestea să fie unice.

Dacă acesta este un lucru înțelept rămâne de văzut, dar puteți să rețineți că zona de nume XSL indicată de Microsoft, `"http://www.w3.org/TR/WD-xsl"`, nu are nici o legătură cu zona de nume actuală a W3C definită pentru standardul XSL. Plasarea de informații existențiale sau sintactice despre anumite implementări ale oricărui standard într-un cod pare plină de riscuri deoarece actualizările oportune sunt imposibile. Pare mult mai probabil că un mecanism viitor va permite codificarea acestei informații într-un anume tip de fișier disponibil pentru URI-ul zonei de nume și regăsit dinamic, permițând actualizarea standardelor la o locație centrală, fără ca informația să fie încapsulată și fosilizată prin codare hard în fiecare aplicație.

Studii de caz

În acest capitol am studiat conceptul de zonă de nume, o metodă de a permite DTD-urilor XML să fie combinate sigur într-un singur document. Am cercetat alternative la zonele de nume pe care să vă puteți baza în situațiile în care suportul pentru zonele de nume nu este disponibil, cum ar fi atunci când se folosesc agenți utilizator moșteniți, care nu suportă acest concept. Astfel de exemple pot fi browserele SGML, care nu cunosc nimic despre zonele de nume sau browserele HTML moștenite, care pot accepta XML numai prin adăugarea sau înlocuirea etichetelor HTML sau XHTML pentru a adăuga sau înlocui documentul XML nativ.

Cu toate că am văzut că zonele de nume XML reprezintă o soluție universală în multe situații, ele nu sunt o soluție ideală. În momentul de față, ele sunt complexe, greu de citit și nu prea potrivite pentru construirea de pagini dinamice utilizând multiple surse XML, chiar dacă transcluderea dinamică a multiple documente XML este implicată de facilitățile XLink și XPointer, discutate în Capitolul 9.

Pare evident că în cele din urmă se va ține cont în mod automat de zonele de nume atunci când mai multe documente sunt citite de pe Web într-un singur document gazdă, cu toate că nici un agent utilizator actual, cu suport XML, nu acceptă un asemenea comportament.

Presiunea cuprinderii transcluderilor dinamice și afișarea de documente aleatoare vor duce în cele din urmă la apariția unor astfel de facilități, deși gama largă de standarde XML și bazate pe XML probabil va face ca suportul complet pentru toate caracteristicile posibile să fie dificil de realizat de către furnizori, în afara unei abordări incrementale.

Am discutat de asemenea câteva dintre dificultățile inerente ale identificatorilor de zone de nume bazați atât pe URL cât și pe URN. Nici unul nu este perfect și pot să ceară o schimbare în concepția proiectanților Web, referitoare la localizarea lucrurilor pe Web. Una dintre tehnologiile posibile care ar putea ușura ambele probleme este Persistent URL, PURL, care interpune un agent sau un server între utilizator și locația finală a unei resurse, astfel încât modificările de locație ale resurselor să fie realizate pe unul sau mai multe situri centrale (care nu se modifică). Puteți afla mai multe informații despre PURL la `http://purl.oclc.org`.

Dar utilizarea completă a zonelor de nume poate să ceară o modificare fără precedent a modului în care sunt finanțate autoritățile de denumire. Dacă o autoritate de denumire devine o parte a sistemului comun XML, fiind utilizată ca sursă de autoritate URN, atunci va fi necesară o metodă de alocare automată a unor stimulente financiare, fie printr-o „taxă" generală (și foarte mică) pe traficul Internet, fie prin alte mijloace. Modelul actual al ASCAP și al altor case de discuri, prin care autorii de muzică sunt plătiți automat pentru utilizarea muzicii lor în sistemul de transmisie radio „gratuită", susține că acest lucru nu ar fi foarte greu, este doar complicat de implementat fără ca părțile să se simtă exploatate în mod necinstit.

Dezvoltarea Web-ului bazat pe XML poate lua multe forme – de fapt, toate formele posibile se vor bate destul de mult timp pentru dominarea pieței – dar putem anticipa direcția probabilă a dezvoltării, prin analogie cu sistemele actuale de radio și televiziune. Atât pay-per-view cât și sistemele finanțate prin publicitate vor înflori, în timp ce multe servicii populare prin abonamente vor fi probabil grupate în echivalentul Web al televiziunii prin cablu, de furnizori dispuși să negocieze pachete mari realizând astfel economii pentru consumatori.

Scheme XML

CAPITOLUL 7

În acest capitol

Scheme XML
Argumente pentru un nou limbaj de scheme
Analiza schemelor XML
Conlucrarea între aptitudinile și posibilitățile schemelor
Codarea unui exemplu simplu de schemă XML
Propunerile anterioare de scheme XML
Depanarea schemelor XML

Scheme XML

Schemele reprezintă generația următoare de limbaje de definire a structurilor, înlocuind sau extinzând mecanismele brute de scheme și tipuri de date încorporate chiar în DTD-ul XML 1.0. Cerințele tipice ale bazelor de date pentru introducerea de date structurate sunt mult mai riguroase decât poate garanta XML 1.0 de unul singur. În timp de XML permite introducerea aproape oricărei informații, de orice lungime și în orice limbă din lume, într-un câmp `CDATA`, este posibil ca o bază de date să accepte numai 20 de caractere ASCII în câmpul intern corespunzător. Fără aptitudinea de a defini lungimea și natura informației introduse în câmpuri de date XML, aplicația trebuie să trimită datele la destinație și să se bazeze pe faptul că greșelile vor fi depistate prin verificarea erorilor și limitelor la destinație. Evident, aceasta ar genera trafic suplimentar pe rețea și, dacă informația trebuie livrată sub formă de pachete, ar cere un decalaj de timp intolerabil între introducerea datelor și aflarea faptului că o anumită porțiune a lor conține erori.

Schemele XML sunt o soluție radicală. Schemele permit verificări reale de limite și erori astfel încât înregistrările XML structurate posibile pot fi analizate și verificate nu numai din punctul de vedere al construirii corecte dar și al validității interne. Din păcate, specificațiile XML Schema actuale sunt printre cele mai dificile și mai obscure dintre toate specificațiile XML. Cu toate că acestea probabil că se vor modifica întrucâtva atunci când vor apărea recomandările finale, lucrurile nu au pornit chiar atât de bine. Grupul de lucru a emis un ghid al schemelor împreună cu recomandările, pentru a ușura întrucâtva procesul de învățare a modului de utilizare a schemelor, dar acest capitol explică mai simplu ceea ce se întâmplă.

În acest capitol veți studia și implementa exemple din ultima schiță de lucru a XML Schema existentă la momentul apariției acestei cărți. Cu toate că nu există implementări efective disponibile pentru a testa codul nostru, puteți simula ieșiri exemplu din versiuni ipotetice cu suport XML ale browserelor existente, pentru a lămuri cum va arăta această tehnologie în practică.

Argumente pentru un nou limbaj de scheme

Deoarece XML este proiectat pentru a fi formatul universal pentru documentele și datele structurate de pe Web, rezultă că una dintre cerințele pentru o facilitate de tip schemă este aceea că utilizează XML ca mediu de exprimare. Acesta este un efort a W3C pentru a constrânge proliferarea fără sfârșit a limbajelor și mecanismelor de sintaxă pe Web, care a permis construcții ad-hoc cum ar fi cele pentru limbajele de scriptare in-line și diversele variante de foi de stil în cascadă care să utilizeze sintaxe și formate de informații complet diferite chiar față de HTML. Indirect, aceasta a condus la practici discutabile cum ar fi transformarea în comentariu a script-ului sau a foii de stil pentru a evita blocarea multor browsere.

Aptitudinea de a transforma în comentariu o foaie de stil sau un script este absentă din punct de vedere teoretic în HTML deoarece conținutul elementelor de stil sau script se presupune a fi `#CDATA`, adică text care nu conține nici un fel de marcaj, nici măcar comentarii. Browserele care știu că aceste două elemente au un conținut de tip `#CDATA`, ce interzice browserului să analizeze datele încadrate în vederea căutării unor marcaje, vor citi textul încadrat, inclusiv comentariul, ca text semnificativ.

Aceasta înseamnă că limbajul de scriptare sau procesorul de foi de stil trebuie să ignore șirul de început de comentariu SGML/XML (`<!--`) pentru a evita blocarea script-ului. Dar șirul de încheiere a comentariului SGML/XML (`-->`) poate încurca procesorul script-ului sau al foii de stil, deci trebuie să transformați în comentariu șirul de încheiere al comentariului SGML, folosind sintaxa de comentariu corespunzătoare limbajului foii de stil, respectiv limbajului de scriptare.

Totuși, browserele și mai vechi, care nu recunosc deloc elementele de script și de stil, vor vedea în mod prestabilit conținutul lor ca `#PCDATA`. La analizarea conținutului, aceste browsere mai vechi vor vedea comentariile și vor ignora întregul conținut încadrat. Este într-adevăr destul de inteligent.

Deci printr-un proces de conexiune logică circulară, browserele moștenite care nu recunosc script-uri tratează conținutul etichetelor de stil și de script ca `#PCDATA` deoarece nu cunosc o metodă mai bună, iar autorii pot încadra conținutul în comentarii pentru a preveni redarea lui de către browser. Browserele care întâmplător cunosc ce înseamnă de fapt elementele de stil și de script *trebuie* să trateze conținutul ca `#CDATA`, deoarece se va ține cont de *toți cei* care transformă în comentariu conținutul script-ului sau al foii de stil, și nu de cei care nu fac asta, deci sunt forțate să ignore comentariul și să vadă de fapt codul inclus în acesta. Este dezordonat chiar dacă funcționează. Dar neglijența întregii metode îi scoate din minți pe proiectanții de limbaje.

Replica oficială este că de pe server ar trebui să utilizați depistarea browserului și să oferiți pagina corespunzătoare pentru fiecare browser. Aceasta vă permite să eliminați complet script-ul sau foaia de stil pentru browserele care nu au suport și să includeți abil script sau foaie de stil cu cod încadrat pentru browserele care le acceptă. Evident, minuțioasa

schemă de comentarii este redundantă dacă procedați astfel. Dar există sute de browsere, iar în curând vor fi și mai multe. Manevrarea rațională a tuturor browserelor este imposibilă pentru mulți și foarte dificilă pentru toți. Nu oricine care are un sit Web are și acces la server.

Singura metodă de a depista browserul în cazul în care nu puteți controla comportamentul serverului este cu un script client-side, dar trebuie ca script-ul să îl transformați în comentariu pentru a evita blocarea browserelor moștenite. Este o dilemă. Deci, CSS-urile in-line și script-urile vor rămâne utile, după cum se pare, în XML și este probabil ca tensiunea dintre browserele moștenite și dorințele proiectanților de limbaje să nu dispară prea curând. Iar existența conținutului util în comentarii va rămâne o problemă iritantă pentru o bună bucată de vreme.

Notă

Recomandarea XML 1.0 precizează că analizoarele XML nu sunt obligate să transfere comentarii spre aplicație. Se consideră că script-urile in-line și foile de stil se află foarte aproape de punctul de a ieși din circulație atunci când sunt vizualizate cu un analizor XML. Secvența de script tipică în HTML utilizează de fapt două mecanisme distincte de comentare pentru a ascunde fiecare set de comentarii unul de celălalt, primul, un comentariu SGML `<!-- -->`, pentru a ascunde codul script-ului de browserele moștenite, iar apoi comentariul JavaScript, `//`, pentru a împiedica script-ul să citească sfârșitul comentariului SGML:

```
<SCRIPT><!-- date script // --></SCRIPT>
```

Faptul că script-urile și foile de stil conțin `#CDATA` înseamnă, în primul rând, că acele presupuse comentarii nu ar trebui analizate ca și comentarii. Dar XML analizează aproape *totul*; prezența marcajului în `#CDATA` înseamnă numai că analizorul XML poate să genereze o eroare fatală.

Dacă script-urile și limbajele de programare externe s-ar potrivi puțin mai bine în universul XML, ele ar fi perfect capabile să manevreze tipul validării intrării pentru a cărui realizare au fost proiectate schemele XML. De fapt, deoarece limbajele de programare cu scop general sunt în mod firesc mai puternice decât mecanismele predefinite de scheme, indiferent cât de versatile, utilizând XML Schema în locul unui program particularizat, renunțați la ceva.

În schimbul unei puteri oarecum mai mici, toată lumea câștigă o accesibilitate largă a caracteristicilor de verificare a intrării. Nu oricine are abilitatea de a învăța și utiliza limbaje de programare mai puternice, sau dorința de a petrece timpul necesar pentru a stăpâni acele abilități. Prin utilizarea XML Schema, lucruri destul de comode pot fi făcute aproape după șablon. Deoarece ele utilizează XML ca bază, ritmul învățării nu este îngrozitor de accelerat și deoarece majoritatea operațiilor obișnuite și a tipurilor de date sunt încorporate, le puteți utiliza cu rapiditate pentru a termina lucrul fără prea mult efort.

Rezistența la modificările în schemele XML

Cu toate că ultima variantă a schiței de lucru XML Schema este încă în dezvoltare, nu se preconizează modificări substanțiale până la publicarea recomandării propuse. W3C și-a planificat să publice o Propunere de Recomandare în 2000, ulterior putând fi făcute modificări până la statutul complet de Recomandare. Actualizări periodice vor fi făcute pe situl Web asociat acestei cărți, `http://www.leeanne.com/usingxml/`, pentru a oglindi dezvoltările recente și pentru a extinde utilitatea în timp a acestei referințe.

Atenţie Orice referinţă tehnică la un standard atât de schimbător ca XML Schema are riscul de a-şi pierde valabilitatea. Există un sit Web asociat acestei cărţi care va urmări modificările şi va actualiza exemplele din carte pentru a oglindi ultimele standarde la `http://www.leeanne.com/usingxml/`.

Între timp, pot fi făcute deja unele lucruri utile cu XML Schema, presupunând că pe platforma aleasă de dumneavoastră există un browser care o suportă. Recomandarea finală va îmbunătăţi numai puterea şi expresivitatea unui instrument care chiar şi în stadiul actual este foarte departe de facilităţile disponibile cu uşurinţă acum. În acest capitol veţi studia metode de a limita tipurile de lucruri care pot fi introduse în câmpurile de date şi atribute, utilizând instrumentele preliminare care există acum. În momentul în care citiţi aceste rânduri, evoluţia schiţei de lucru W3C poate forţa regândirea unor părţi ale acestei discuţii, dar temelia acelor gânduri va fi clară. Ca întotdeauna, situl Web al W3C, `http://www.w3.org/`, este arbitrul final al practicii curente recomandate.

Analiza schemelor XML

Au existat mai multe propuneri pentru W3C referitoare la problema schemelor într-o formă sau alta. De fapt, mecanismul DTD XML existent este un fel de schemă. În secţiunea „Propunerile anterioare de scheme XML" din acest capitol, veţi afla câte ceva despre diversele propuneri.

Grupul de lucru W3C, încercând să rezolve toate aceste propuneri contradictorii, a dezvoltat o soluţie de compromis, XML Schema. XML Schema este o extensie compatibilă cu XML 1.0 a limbajului de bază XML 1.0, care permite extinderea aproape infinită a aptitudinilor de constrângere proprii DTD-urilor XML 1.0. În locul tipurilor elementare de conţinut caracter, element sau combinate ale DTD-ului XML 1.0, XML Schema permite, de exemplu, limitarea conţinutului la un şir de caractere de o anumită lungime sau la un întreg dintr-un interval specificat, permiţând şi o reglare mult mai fină a modelului de conţinut. De asemenea, are posibilitatea de a elimina nevoia de utilizare a unui limbaj special pentru DTD, deoarece pentru definirea tipurilor de date utilizează chiar XML.

Punctele forte ale schemelor

Chiar şi în forma sa actuală, XML Schema are perspective uriaşe. Este un mecanism flexibil de grupare a atributelor, analog cu declaraţia de entitate parametru din DTD-urile XML, dar mult mai adaptabil, cu aptitudinea de a declara noi tipuri de date, o facilitate de modelare care vă permite să creaţi structuri care partajează în mod vizibil componentele lor comune şi multe altele.

Pot fi declarate valori minime şi maxime pentru elementele de date, într-un singur document pot fi utilizate mai multe scheme, iar conceptul validităţii schemelor se dezvoltă rapid.

Schiţa de lucru actuală vă permite de asemenea să specificaţi un *model* pentru elemente şi atribute care utilizează sintaxa obişnuită a *expresiilor regulate* pentru a descrie genul de lucruri care sunt permise într-un anumit câmp. Modelul expresiei regulate ţintă este

cel Unicode, care are câteva extensii minore faţă de expresiile regulate utilizate în shell-ul UNIX, în numeroase editoare şi programe pentru e-mail profesionale şi în altele. De exemplu, pentru a specifica un element cu un model, veţi crea un tip nou bazat pe un tip existent şi veţi defini modelul astfel:

```
<simpleType name='ZipCode' base='string'>
  <pattern value='\d{5}'/>
</simpleType>
```

Acest model va defini un cod ZIP de cinci cifre, aşa cum este utilizat acesta de Serviciul Poştal din SUA pentru sortarea corespondenţei.

Sugestie de la LeeAnne.com Words to weave by

O referinţă bună la expresiile regulate în stil Unicode se găseşte la `http://www.unicode.org/unicode/reports/tr18/`. Unicode necesită extensii semnificative la expresii regulate în stil UNIX pentru a manevra secvenţe de caractere multiple atunci când acestea se comportă ca un singur caracter în unele limbi. Exemplele includ „ch" sau „ll" din spaniolă, care se comportă ca o singură literă atunci când sunt citite. Extensiile sunt necesare, de asemenea, pentru a permite specificarea caracterelor prin nume mnemonice în locul valorilor numerice.

În combinaţie cu alte facilităţi XML, chiar şi acum, în procesul de dezvoltare, XML Schema adresează o parte importantă a problemei noastre, lăsând mult mai puţin de rezolvat în seama designerilor sau programatorilor Web supraîncărcaţi.

Identificarea compromisurilor schemelor

Pe de altă parte, schiţa de lucru XML Schema existentă (la momentul scrierii acestei cărţi) reprezintă un pas înapoi de la internaţionalizare, spre o orientare centrată pe SUA, numai în engleză şi în stil ASCII. În acest moment, numerele utilizează numai simboluri europene şi nu există suport pentru alte simboluri Unicode, chiar dacă acestea sunt incluse în specificaţia XML 1.0.

Singurul calendar acceptat este calendarul gregorian laic, deşi a fost extins standardul ISO-8601 pe care se bazează calendarul şi sistemul de timp al XML Schema, pentru a permite atât ani negativi, cât şi ani mai mari decât 9999. Deci, lumile străvechi şi cele foarte îndepărtate în viitor pot fi reprezentate (măcar aproximativ şi numai în forma gregoriană), dar orice altceva trebuie suportat prin algoritmi, inclusiv datele calendaristice în sistem iulian şi în alte sisteme calendaristice care nu sunt gregoriene. Sărbătorile creştine variabile (de exemplu, Paştele), majoritatea sărbătorilor care nu sunt creştine, multe dintre ele utilizând un calendar lunar (islamic) sau lunisolar (chinezesc/ebraic) şi chiar împărţirea anului în săptămâni şi numele zilelor, nu sunt disponibile în schiţa actuală. Evident, există algoritmi binecunoscuţi pentru manevrarea tuturor acestor conversii, dar sunt atât de multe încât pot să înţeleg dificultatea includerii tuturor într-un singur standard.

Punctul zecimal necesar măcar într-o porţiune a specificaţiei este punctul latin utilizat peste tot în Statele Unite, cu toate că, pentru acelaşi scop, în multe ţări europene se foloseşte virgula. Aptitudinea comodă şi uzuală de a delimita miile cu un marcaj de diviziune (fie o virgulă, fie un punct în utilizarea vestică) este de asemenea absentă.

Nici una dintre aceste restricţii nu poate depăşi stadiul de schiţă de lucru. Au fost adăugate deja expresiile regulate în stil Unicode care permit multe în privinţa verificării erorilor şi rezolvă complet cele două probleme prezentate mai sus. Este posibil ca XSL şi XSLT

să poată manevra mult mai multe. Deci este prematur să spunem că o idee sau alta nu este prea bună. De fapt, standardul este influențat de deciziile luate în alte grupuri de lucru, de a căror muncă depinde. Este o sarcină dificilă și ingrată, iar eu personal nu voi învinovăți pe nimeni de nimic.

Sper ca atributele și alte mecanisme sintactice pe care se bazează XML Schema să nu fie exprimate într-o engleză atât de tehnică încât să se supună jargonului programatorilor.

Totuși, pare posibil ca această construire a schemelor, chiar mai mult decât cea a DTD-urilor, să rămână în domeniul specialiștilor din toată lumea. Cu toate că aceasta reflectă concentrarea de putere și tehnologie din lume, este păcat că nu a fost creată o metodă neutră de specificație a schemelor care să facă tehnologia mai accesibilă pentru aceia care utilizează și sunt mai familiarizați cu sistemele de scriere non-europene. Dar standardele W3C doresc să fie mai mult practice decât ideale. Inventarea unui instrument pentru toate limbile poate depăși totuși capacitățile acestor servitori credincioși ai binelui comun.

Conlucrarea între aptitudinile şi posibilităţile schemelor

Cel puțin în momentul actual, schița de lucru XML Schema este împărțită în două componente principale, *XML Schema Part 1: Structures* (structuri) și *XML Schema Part 2: Datatypes* (tipuri de date) și, în plus, *XML Schema Part 0: Primer* (abecedar) pentru a vă ajuta să înțelegeți totul. Ele sunt adresate aici în combinație, deoarece în practică este nevoie de ambele pentru a face un lucru util. Evoluția produsului este rapidă, deci această secțiune vorbește despre nevoile de care este conștient W3C și face câteva presupuneri referitoare la direcția spre care se îndreaptă.

Utilizarea zonelor de nume ale schemelor

XML Schema utilizează zonele de nume ușor diferit de restul setului de standarde XML, iar dezvoltările sale vor pătrunde inevitabil și în restul standardelor XML înrudite. În ciuda faptului că partea despre zonele de nume a specificației poate să se modifice puțin în drumul spre statutul final de recomandare, în momentul de față intenția este clară. Se încearcă să se ofere mecanisme de rezoluție alternative, iar unele sau toate acestea vor fi folosite de un proiectant de aplicații cu suport XML Schema pentru a găsi fișierele schemă potrivite, în funcție de zonele de nume. În toate cazurile, se așteaptă ca aplicația să *dereferențieze* (un termen de programare precis pentru prelucrarea și transcluderea unui pointer sau a unei hiperlegături) URI-ul uneia sau al mai multor zone de nume XML speciale pentru a permite, oricât de indirect, uneia sau mai multor scheme să refere schemele corespunzătoare pentru fiecare zonă de nume din document. Există posibilități diverse pentru găsirea schemei potrivite, care sunt listate aici într-o ordine aleatoare. Specificația XML Schema nu cere unei aplicații să implementeze o anumită strategie de căutare pentru a găsi o schemă potrivită și nici măcar să caute o schemă XML dacă nu este nevoie:

- O opțiune este să nu faceți nimic. Dacă aplicația știe deja unde este localizată schema, ea nu trebuie să aplice metode euristice pentru a o găsi. În același mod, dacă aplicația știe că nu poate fi găsită nici o schemă deoarece informația este garantată ca fiind compatibilă la intrare cu schema, nu trebuie testată nici locația și nici validitatea schemei. În mod alternativ, aplicația poate să aibă cunoștințe interne despre schema relevantă și poate testa validitatea fără a face referiri la o schemă externă.
- O altă opțiune permite aplicației să caute o schemă în funcție de locația documentului. Poate fi găsită ca un fișier schemă local efectiv, un articol într-un depozit sau cache de scheme locale, sau ca o referință indirectă într-un fișier catalog local.
- Aplicația poate căuta o schemă în funcție de zona de nume a documentului sau de orice zonă de nume asociată elementelor și atributelor documentului. O schemă poate fi găsită ca un fișier schemă local, un articol într-un depozit sau cache de scheme locale, sau ca o referință indirectă într-un fișier catalog local.
- Aplicația poate căuta locația efectivă a documentului pe Web, spre deosebire de utilizarea locației ca parametru într-un algoritm de căutare local. Din acel punct, strategia de căutare poate fi aplicată ca și când ar fi locală.
- Aplicația poate căuta zona de nume efectivă a documentului și orice zonă de nume asociată cu elementele și atributele documentului pentru un fișier schemă, spre deosebire de utilizarea locației ca parametru într-un algoritm de căutare local. Din acel punct, strategia de căutare poate fi aplicată ca și când ar fi locală.
- Deoarece metodele euristice anterioare pot găsi mai multe scheme, acele scheme pot considera oportună încorporarea atributelor `nsi:targetNamespace` în declarațiile lor, pentru a ajuta aplicația să determine care scheme vor fi corespunzătoare.

Zonele de nume Schema pot fi găsite și prin indirectarea serverului, deoarece serverul asociat fie documentului însuși, fie zonei sale de nume, este posibil să aibă informații despre locația predefinită a schemei relevante în interiorul său. Deoarece locațiile și zonele de nume pot conține URN-uri cu o locație asociată cunoscută numai de o autoritate centrală, găsirea locației efective poate dura mai mult. Din acest motiv, specificația schemelor stabilește în prezent că dereferențierea poate fi eliminată la cererea utilizatorului sau prin proiectare în sistemele înglobate unde se știe dinainte că toate documentele sunt în concordanță cu schema.

Utilizați una sau mai multe dintre procedurile următoare pentru a crea una sau mai multe referințe XML Schema pentru fiecare zonă de nume asociată documentului:

Deoarece XML poate încorpora într-un singur document părți ale mai multor documente și zone de nume document, XML Schema poate asocia explicit o schemă cu fiecare zonă de nume pentru a accelera procesul de căutare. Pentru a realiza acest lucru, trebuie să declarați zona de nume în modul obișnuit, să declarați zona de nume XML Schema specială și apoi să specificați din nou URI-ul zonei de nume împreună cu locația documentului schemă care se referă la el, astfel:

```
<documentname xmlns:doc="http://www.leeanne.com/usingxml/"
              xmlns:htm="http://www.w3.org/1999/XHTML"
              xmlns:xsi="http://www.w3.org/1999/XMLSchema/instance"
      xsi:schemaLocation="http://www.leeanne.com/usingxml/
                          http://www.leeanne.com/doc.xsd
                          http://www.w3.org/1999/XHTML
                          http://www.leeanne.com/usingxml/xhtml.xsd">
```

Dacă trebuie să găsiți o schemă care nu are predefinită `xsi:targetNamespace`, puteți utiliza atributul `xsi:noNamespaceSchemaLocation` pentru a identifica o anumită locație, astfel:

```
xsi:noNamespaceSchemaLocation="http://www.leeanne.com/usingxml/">
```

În acest punct, oricare dintre metodele euristice prezentate anterior poate fi utilizată

pentru a localiza schema efectivă, cu excepția faptului că ultimul aspect referitor la atributul `xsi:targetNamespace` nu va fi aplicat.

Documentele schemă în sine pot fi alcătuite din părți. Pentru a include o altă schemă sau o porțiune dintr-o singură zonă de nume, procedați astfel:

```
<include schemaLocation = uri-reference />
```

Pentru a include o schemă aflată în relație cu o anumită zonă de nume, importați-o cu o zonă de nume și cu o pereche de scheme de zone de nume, astfel:

```
<import namespace="URI" schemaLocation="URI" />
```

Mai există încă unele probleme cu zonele de nume în XML Schema. CSS și multe alte aplicații XML au (sau nu) mecanisme diferite pentru rezolvarea zonelor de nume multiple în documentele incluse, mai ales atunci când aceeași foaie de stil poate fi utilizată pentru a referi mai multe zone de nume. Sintaxa utilizată pentru zone de nume multiple în XML Schema nu se extinde la documentele care o utilizează, deci CSS sau foile de stil XSL aplicate unui document XML utilizând XML Schema este posibil să nu funcționeze corect, în afară de cazul în care multiplele utilizări actuale ale zonelor de nume sunt reglementate și sunt consecvente în toate standardele XML.

Notă Zonele de nume în scheme se tot modifică. Acum, pare probabil că va exista un fișier schemă fizic la URI-ul indicat de fiecare declarație de zonă de nume, dacă este prezent un URI. Pe de altă parte, o zonă de nume inaccesibilă sau abstractă, identificată de un URN (Uniform Resource Name), cum ar fi un număr ISBN unic, nu va avea o locație fizică, deci o schemă va trebui descoperită prin alte mijloace.

Deoarece aceasta implică la rândul său și eventuala nevoie de acces la Internet, se pare că va fi nevoie să meargă mână în mână cu validarea DTD-ului codului XML în sine. Poate va fi necesar accesul prin zona de nume și la alte forme de date externe, poate DTD-uri, deși schemele XML pot să facă chiar ele aproape tot ceea ce este asociat unui DTD. Ceea ce am spus mai sus ar deschide drumul unor complicații și mai mari dacă nu ar fi doar o hiperbolă, deși în principiu are sens.

Dacă validați câmpurile unui document XML, definite prin scheme, nu prea are sens să nu validați documentul ca întreg. Dacă acele câmpuri și acel document sunt folosite pentru a valida un ecran pentru introducere de date, nu are nici un sens să permiteți unui analizor care nu validează să afișeze câmpurile de intrare. Dar XML nu definește actualmente o modalitate de a forța validarea unui document XML. În plus, un procesor care nu validează este definit acum numai în relație cu un DTD, însă o schemă XML înlocuiește complet DTD-ul. Un analizor poate fi numit în mod legitim „care nu validează" pentru că nu a validat documentul față de DTD, permițând în același timp documentului să fie validat față de o schemă XML. Dată fiind existența unei scheme XML care poate, teoretic, să înlocuiască DTD-ul, conceptul de analizor care nu validează devine neclar și greu de definit.

Validarea intrărilor nu pare să merite efortul dacă utilizatorii pot evita necesitatea supunerii intrării lor la o cercetare atentă alegând pur și simplu un alt software. De asemenea, pot să apară probleme de securitate. Deoarece sunt disponibile mai multe zone de nume, fiecare dintre acestea putând să redefinească în întregime o schemă calificată pentru zona de nume, problema interacțiunii acestora în același document nu are o rezolvare ușoară. Sunt multe probleme de rezolvat înainte ca schemele XML și în special zonele de nume să ajungă în prim-plan.

Între timp, după cum veți vedea în Capitolul 14, „Utilizarea generației următoare de foi de stil extensibile – XSL", este posibil să utilizăm XSLT (Extensible Style Sheet Language Transformations – limbajul pentru transformarea foilor de stil extensibile) pentru a soluționa parțial multe dintre problemele care au rămas de rezolvat pentru W3C în momentul de față, în loc să așteptăm ca W3C să rezolve toate problemele noastre.

Codarea unui exemplu simplu de schemă XML

Listingul 7.1 furnizează un exemplu prescurtat de document XML care utilizează XML Schema pentru a valida structurile de elemente și câmpurile de date.

Listingul 7.1 Un exemplu XML de înregistrare a clienților

```
<?xml version="1.0"?>
<!-- Customer Record -->
<customer xmlns="http://www.leeanne.com/aristotelian/"
          xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.leeanne.com/aristotelian/schemas"

"http://www.leeanne.com/aristotelian/schemas/07xmp01.xsd">
  <customerName>Aristotelian Logic Systems</customerName>
  <customerID>112344-0897</customerID>
  <credit-status>Net-30</credit-status>
  <customer-contact locale="USA">
    <honorific>Ms.</honorific>
    <given-name1>George</given-name1>
    <given-name2>Evelyn</given-name2>
    <family-name>Fayne</family-name>
    <degrees>LLD, PhD</degrees>
    <title>Contracts Director</title>
    <department>Purchasing</department>
    <telephone locale=NANP">+1 101 555.1234</telephone>
    <extension>3456</extension>
  </customer-contact>
  <comment>ALS buys over $300K per year, primarily our Mark IV WKS</comment>
  <shipping-address locale="USA">
    <attention>Attn:</attention>
    <attention-name>Receiving Clerk</attention-name>
    <reference>Ref:</reference>
    <street-address1>987 Technology Pkwy</street-address1>
    <street-address2>Unit 200 - Receiving</street-address2>
    <city>Riverdale</city>
    <territorial-division>Ohio</territorial-division>
    <territorial-abbreviation>OH</territorial-abbreviation>
    <postal-code>31089</postal-code>
    <country>USA</country>
  </shipping-address>
  <billing-address locale="USA">
    <attention>Attn:</attention>
```

```
    <attention-name>Accounts Payable</attention-name>
    <reference>Ref:</reference>
    <street-address1>987 Technology Pkwy</street-address1>
    <street-address2>Unit 300 - Accounting</street-address2>
    <city>Riverdale</city>
    <territorial-division>Ohio</territorial-division>
    <territorial-abbreviation>OH</territorial-abbreviation>
    <postal-code>31089</postal-code>
    <country>USA</country>
  </billing-address>
  <creation-date>1995-07-11</creation-date>
  <last-updated>1999-10-11</last-updated>
</customer>
```

În Listingul 7.1 există atribute locale în liniile 10, 18, 22 și 34, care permit schemei să valideze aceste câmpuri în funcție de diferite criterii depinzând de locația introdusă în acest atribut.

Observați mai ales utilizarea facilității XML Schema `noNamespaceSchemaLocation` pentru a indica schema efectivă utilizată pentru a valida acest document conform schemei, spre deosebire de validarea lui față de un DTD. Această locație, care poate conține de fapt o listă de locații trecute fie ca directoare, fie ca nume de fișiere efective, va fi dereferențiată pentru a rezulta o schemă efectivă dacă nu poate fi găsită o schemă la zona de nume a documentului în sine. În toate situațiile, construiți arborele documentului XML și apoi parcurgeți arborele în preordine, căutând elemente sau atribute cu definiții de schemă accesibile, bazate pe zonele lor de nume. Când sunt găsite astfel de elemente sau atribute, validați față de schemă și transformați arborele documentului în consecință. Acest gen de căutare recursivă ad-hoc a siturilor Web externe prezintă o mare dificultate pentru navigarea în timp real.

Definiția schemei

În acest exemplu, mai multe câmpuri sunt marcate cu atributul `locale` care le identifică din punctul de vedere al locului lor, deci poate fi realizată un fel prelucrare specială a lor până când specificația XML Schema a W3C va ajunge din urmă nevoile lumii reale. Cu toate că acest lucru poate fi făcut cu DOM (Document Object Model – modelul document obiect) și cu orice limbaj capabil să analizeze și să acționeze în arborele documentului, validarea va trebui făcută de dumneavoastră. Deoarece XML Schema are încorporate aceste facilități, nu este necesar să reinventați roata. În Capitolul 14 veți vedea cum puteți utiliza XSL (Extensible Style Sheet Language – limbajul extensibil al foilor de stil) pentru a transforma un document XML obișnuit într-o versiune localizată care poate fi validată cu ușurință față de o singură schemă.

Atribute `locale`

Majoritatea atributelor `locale` din acest exemplu sunt de tipul `USA`, deoarece aici este localizat clientul, dar numărul de telefon este marcat ca aparținând NANP (North American Numbering Plan – planul de atribuire a numerelor în America de Nord), care acoperă Statele Unite, Canada și o parte din Caraibe.

Acest ultim aspect se datorează faptului că aceste națiuni folosesc același model de apelare telefonică, spre deosebire de restul lumii care utilizează coduri de acces „speciale" și

numere de telefon având un aspect „ciudat" din punctul de vedere al nord-americanilor. Pentru a permite validarea și, eventual, utilizarea acestei înregistrări cu un sistem de formare automată a numărului de telefon, aveți nevoie de o metodă pentru a deosebi numerele de telefon din țări diferite.

De asemenea, identificați informațiile de contact ca fiind din Statele Unite, deoarece utilizarea titlurilor și a numelor de organizații diferă de la o țară la alta, iar numărul de telefon poate să nu reflecte naționalitatea și preferințele de formatare a numelui abonatului. Mulți chinezi, de exemplu, preferă ca numele de familie să apară primul, urmat de prenume. Acest lucru nu poate fi făcut din punct de vedere algoritmic dintr-un simplu câmp nume deoarece unele nume de familie chinezești sunt formate din două caractere și nu vă puteți baza pe faptul că acele persoane vor introduce, fără ajutor, numele în ordinea corectă.

Ca un exemplu de prelucrare diferită a câmpurilor, bazată pe `locale` (sau pe orice alt atribut), tipul `USA` din adresă vă permite să validați formatul codurilor ZIP, de exemplu, care sunt formate fie din cinci cifre fie din cinci cifre, o cratimă și apoi patru cifre. Un cod poștal canadian utilizează o literă, o cifră, o literă, un spațiu, o cifră, o literă și apoi o cifră – ceva de genul M5A 4N7 – deci o schemă de validare S.U.A. nu va funcționa. În Marea Britanie, întâlniți coduri poștale similare (dar nu complet) care pot arăta astfel: E17 4DZ (o literă urmată de două cifre, un spațiu și apoi o cifră urmată de două litere).

Sugestie de la LeeAnne.com Words to weave by

Validarea convențiilor umane este dificilă pentru un calculator. Soluția de rezervă, să nu validați deloc, ar trebui luată în considerare întotdeauna pentru câmpurile care nu au o importanță prea mare. Cui îi pasă dacă o persoană introduce un număr de telefon incorect? Mulți oameni nu au telefon. Dar dacă numărul asigurării sociale este incorect, aceasta poate avea repercusiuni legale.

După cum puteți vedea, validarea tipurilor de date dă de gândit pe piața internațională. În baza de date subiacentă pot să nu existe restricții asupra tipului de date care pot fi conținute în aceste câmpuri, deoarece validarea potrivită poate fi determinată numai din context.

În exemplu, atributele `locale` furnizează acest context, indicând anumite versiuni care conțin variante ale aceluiași câmp de date, cu toate că nu puteți face asta deocamdată. Între timp, toate câmpurile sunt definite în termenii cei mai generali, ca șiruri, cu versiunile care conțin variante așteptând în culise, ca să zicem așa.

Priviți schema prezentată în Listingul 7.2, care este asociată cu Listingul 7.1 din secțiunea precedentă, luând în considerare capcanele în care este posibil să cădeți dacă nu păsiți cu atenție. Acesta arată întregul, înainte de a-l desface în părți componente. Pentru simplitate, această schemă este prezentată ca o înregistrare monolitică formată dintr-o colecție de tipuri complexe. Listingul prezintă tipul înregistrării client care poate fi o parte a unei baze de date cu clienții unei firme, prezentând toate datele care pot fi necesare pentru departamentele de expediere, contabilitate și vânzări.

Listingul 7.2 Un exemplu de schemă XML pentru înregistrarea clienților

```
<schema name="http://www.leeanne.com/aristotelian/schema/"
        xmlns="http://www.w3.org/1999/XMLSchema">
  <element name="customer">
```

```
    <complexType>
      <sequence>
        <element name="customerName" type="string" />
        <element name="customerID" type="string" />
        <element name="credit-status" type="string" />
        <element name="customer-contact">
          <complexType>
            <sequence>
              <element name="honorific" type="string" />
              <element name="given-name1" type="string" />
              <element name="given-name2" type="string" />
              <element name="family-name" type="string" />
              <element name="degrees" type="string" />
              <element name="title" type="string" />
              <element name="department" type="string" />
              <element name="telephone">
                <complexType base="string" derivedBy="extension">
                <attribute name="locale" type="string" />
                </complexType>
              </element>
              <element name="extension" type="string" />
            </sequence>
            <attribute name="locale" type="string" />
          </complexType>
        </element>
        <element name="comment" type="string" />
        <element name="shipping-address">
          <complexType>
            <sequence>
              <element name="attention" type="string" />
              <element name="attention-name" type="string" />
              <element name="reference" type="string" />
              <element name="street-address1" type="string" />
              <element name="street-address2" type="string" />
              <element name="city" type="string" />
              <element name="territorial-division" type="string" />
              <element name="territorial-abbreviation" type="string" />
              <element name="postal-code" type="string" />
              <element name="country" type="string" />
            </sequence>
            <attribute name="locale" type="string" />
          </complexType>
        </element>
        <element name="billing-address">
          <complexType>
            <sequence>
              <element name="attention" type="string" />
              <element name="attention-name" type="string" />
              <element name="reference" type="string" />
              <element name="street-address1" type="string" />
              <element name="street-address2" type="string" />
```



```
              <element name="city" type="string" />
              <element name="territorial-division" type="string" />
              <element name="territorial-abbreviation" type="string" />
              <element name="postal-code" type="string" />
              <element name="country" type="string" />
            </sequence>
            <attribute name="locale" type="string" />
          </complexType>
        </element>
        <element name="creation-date" type="date" />
        <element name="last-updated" type="date" />
      </sequence>
      <attribute name="locale" type="string" />
    </complexType>
  </element>
</schema>
```

Figura 7.1 prezintă rezultatul validării acestei scheme cu XML Spy de la Icon Systems, la http://www.xmlspy.com/, un mediu de creare XML care acceptă construirea și editarea asistate de scheme, dar și prin XSL și DTD-uri standard.

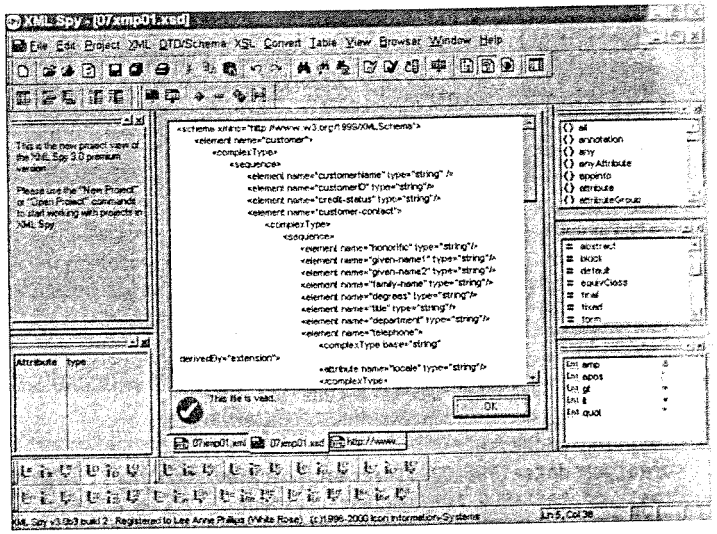


FIGURA 7.1
În XML Spy, înregistrarea client este afișată sub formă de cod XML.

Acum puteți crea un exemplu de înregistrare a clienților utilizând schema ca prototip. Listingul 7.3 prezintă fișierul cod completat.

Listingul 7.3 O înregistrare a clienților având asociată o schemă XML

```
<?xml version="1.0"?>
<!-- Customer Record -->
<customer xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="http://www.leeanne.com/aristotelian/schemas/07xmp01.xsd">
  <customerName>Aristotelian Logic Systems</customerName>
```

```
  <customerID>112344-0897</customerID>
  <credit-status>Net-30</credit-status>
  <customer-contact locale="USA">
    <honorific>Ms.</honorific>
    <given-name1>George</given-name1>
    <given-name2>Evelyn</given-name2>
    <family-name>Fayne</family-name>
    <degrees>LLD, PhD</degrees>
    <title>Contracts Director</title>
    <department>Purchasing</department>
    <telephone locale=NANP">+1 101 555.1234</telephone>
    <extension>3456</extension>
  </customer-contact>
  <comment>ALS buys over $300K per year, primarily our Mark IV WKS</comment>
  <shipping-address locale="USA">
    <attention>Attn:</attention>
    <attention-name>Receiving Clerk</attention-name>
    <reference>Ref:</reference>
    <street-address1>987 Technology Pkwy</street-address1>
    <street-address2>Unit 200 - Receiving</street-address2>
    <city>Riverdale</city>
    <territorial-division>Ohio</territorial-division>
    <territorial-abbreviation>OH</territorial-abbreviation>
    <postal-code>31089</postal-code>
    <country>USA</country>
  </shipping-address>
  <billing-address locale="USA">
    <attention>Attn:</attention>
    <attention-name>Accounts Payable</attention-name>
    <reference>Ref:</reference>
    <street-address1>987 Technology Pkwy</street-address1>
    <street-address2>Unit 300 - Accounting</street-address2>
    <city>Riverdale</city>
    <territorial-division>Ohio</territorial-division>
    <territorial-abbreviation>OH</territorial-abbreviation>
    <postal-code>31089</postal-code>
    <country>USA</country>
  </billing-address>
  <creation-date>1995-07-11</creation-date>
  <last-updated>1999-10-11</last-updated>
</customer>
```

Toate elementele înregistrării din schema prezentată în Listingul 7.3 sunt definite într-o ordine secvențială. Aceasta asigură că schema va impune ordinea corectă a apariției elementelor atunci când va fi utilizată într-un mediu de creare cum ar fi XML Spy sau atunci când va fi validată de un analizor care validează. Pentru elementele de alt tip decât `type="date"`, care utilizează și impune un format de dată calendaristică ISO 8601, acest exemplu utilizează, pentru simplitate, doar validări minimale. În utilizare, ambele date vor fi generate probabil automat, dar aceasta nu o veți ști niciodată.

Figura 7.2 prezintă rezultatul validării documentului XML însuși utilizând schema ca echivalent al unui DTD.

S-ar putea întâmpla ca unele dintre rezolvările despre care veți afla să fie eliminate prin dezvoltările ulterioare ale W3C, dar principiile generale vor rămâne neschimbate: cu suficientă gândire, puteți realiza aproape orice operație arbitrară utilizând instrumentele pe care le aveți la îndemână.

Definirea elementelor

Un element este definit în termenii tipurilor de bază, care pot include un conținut bogat (mixt), tipuri de date sau numai elemente, ca în XML 1.0. Cu toate că sintaxa seamănă cu definiția unui DTD XML, în realitate utilizează sintaxa XML obișnuită, și nu limbajul DTD special. Tipurile de date pot fi extinse sau restrânse prin referirea la un tip de bază într-un fel de moștenire orientată spre obiecte. Deoarece această recomandare poate suferi încă modificări, codul din Listingul 7.4 sugerează numai genurile de atribute disponibile într-un element și tipurile generale pe care le pot conține modelele de conținut ale atributelor. Nu veți studia acest lucru în detaliu, deoarece este încă puțin neclar.

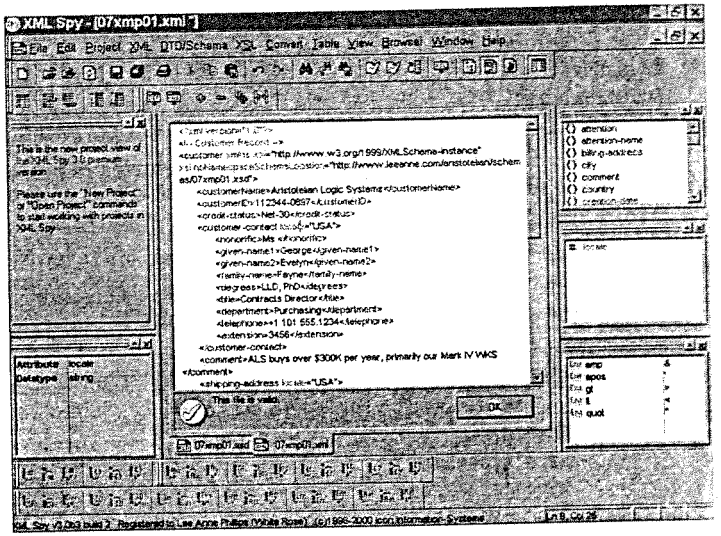


FIGURA 7.2
În XML Spy, înregistrarea clienților validată prin schemă este afișată ca și cod XML.

Listingul 7.4 Structura unui element XML Schema

```
<element name="elementname"
         id="ID"
         ref="typeReference"
         type="typeName"
         minOccurs= 'Integer' | '1'
         maxOccurs= 'Integer' | '*'
         nullable= 'true' | 'false'
         equivClass="className"
         abstract= 'true' | 'false'
         final="setValue" | ''
         block="setValue" | ''
         default="value"
         fixed="value"
```

```
        form="formValues">
  <!-- Content: (( annotation )?,
                ( complexType | simpleType )?,
                ( unique | key | keyref ) -->
</element>
```

Listingul 7.4 reprezintă numai stadiul de gândire actual și este pasibil de modificare. Informațiile cele mai demne de încredere se găsesc întotdeauna pe situl Web al W3C și vor fi urmărite, pentru utilitatea acestei cărți, pe situl meu Web, la `http://www.leeanne.com/usingxml/`. În utilizare, puteți crea structuri simple sau complexe, ca și într-un DTD XML, prin includerea elementelor conținut sau a tipurilor de date complexe în definiția unui element schemă, așa cum este prezentat și în codul următor:

```
<element name="anything">
  <complexType>
    <sequence>
      <element name="One" type="string" />
      <element name="Two" type="string" />
    </sequence>
  </complexType>
</element>
```

Puteți utiliza, de asemenea, un element `choice` pentru a defini selecții alternative dintr-o listă de elemente.

Definirea atributelor

Atributele sunt definite în aproape același mod ca elementele, după cum se vede și în Listingul 7.5.

Listingul 7.5 Structura unui atribut XML Schema

```
<attribute name="name"
           id="ID"
           ref="typeName"
           type="typeName"
           use= 'prohibited' | 'optional' | 'required' | 'fixed' | 'default'
           value="value"
           form="formValues">
<!-- Content: (( annotation )?, ( simpleType )?)  -->
</attribute>
```

Listingul 7.5 reprezintă numai stadiul de gândire actual și este pasibil de modificări însemnate. Informațiile cele mai demne de încredere se găsesc întotdeauna pe situl Web al W3C și vor fi urmărite, pentru utilitatea acestei cărți, pe situl meu Web, la `http://www.leeanne.com/usingxml/`.

Definirea tipurilor de date

O parte a puterii schemelor XML stă în posibilitatea lor de a extinde tipurile de date de bază predefinite în XML Schema prin constrângeri și extinderi. De fapt, cam asta este totul. Abilitatea de a defini câmpuri ca fiind de tipuri primitive predefinite, cum ar fi

numerice, booleene sau caracter, nu este deloc o îmbunătățire a DTD-urilor XML de bază, dar prin extinderea sau limitarea acelor tipuri de bază, în câmpurile dumneavoastră de date pot fi impuse șabloane de o complexitate arbitrară. Tipurile de date extinse pot fi complexe, având mai multe părți, sau simple.

Nivelul de complexitate permis de facilitățile `datatype` ale schemelor XML este extraordinar. Fiecare tip de dată are definite *fațete* care permit ca tipul de dată de bază să fie modificat cu ușurință. Iată un exemplu de număr în notație științifică, care din punct de vedere teoretic poate fi infinit, dar este constrâns la intervalul de valori care poate fi reprezentat în formatul în virgulă mobilă IEEE pe 32 de biți:

```
<datatype name="ieee32">
  <basetype name="float"/>
  <minInclusive>
      1.40239846e-4
  </minInclusive>
  <maxInclusive>
     3.40282347e3
  </maxInclusive>
</datatype>
```

Fațetele pentru acest exemplu sunt elementele `minInclusive` și `maxInclusive` conținute în elementul `datatype`, dar toate fațetele de constrângere funcționează în același mod. Deși dacă știți că veți încărca aceasta într-o bază de date astfel încât acestea să fie limitele câmpului, această constrângere particulară va preveni depășirea superioară sau inferioară.

Lista următoare conține tipurile de date de bază și fațetele, așa cum sunt ele definite actual de către grupul de lucru XML Schema al W3C. Lista poate să sufere modificări.

- **`string`** – Un șir de caractere, după cum este definit de ISO 10646 și Unicode. Are fațete de constrângere: `pattern`, `length`, `enumeration`, `minlength`, `maxlength`, `maxInclusive`, `maxExclusive`, `minInclusive` și `minExclusive`. Ultimele patru fațete depind de ordinea de interclasare, care este un subiect foarte complex în aproape orice limbaj.
- **`boolean`** – O valoare booleană, `true` sau `false`. Are o singură fațetă de constrângere, `pattern`.
- **`float`** – Un număr real. Are fațete de constrângere: `maxInclusive`, `maxExclusive`, `minInclusive`, `minExclusive`, `pattern` și `enumeration`.
- **`double`** – Un număr real de dublă precizie. Are fațete de constrângere: `maxInclusive`, `maxExclusive`, `minInclusive`, `minExclusive`, `pattern` și `enumeration`.
- **`decimal`** – Are aceleași fațete de constrângere ca `float`, `maxInclusive`, `maxExclusive`, `minInclusive`, `minExclusive`, `pattern` și `enumeration`; în plus, are următoarele fațete: `scale` și `precision`. `precision` este numărul de cifre de după punctul zecimal. `scale` este numărul total de cifre alocate tipului de dată.
- **`integer`** – Un subtip evident al lui `decimal`, cu fațetele `scale` și `precision` setate la 0. Cu alte cuvinte, are utilele fațete de constrângere `maxInclusive`, `maxExclusive`, `minInclusive`, `minExclusive`, `pattern` și `enumeration`.
- **`long`** – Un subtip în stil IEEE al lui `integer`, cu `maxInclusive` setat la 9223372036854775807 și `minInclusive` setat la –9223372036854775808.
- **`int`** – Un subtip în stil IEEE al lui `long`, cu `maxInclusive` setat la 2147483647 și `minInclusive` setat la –2147483648.

- **short** – Un subtip în stil IEEE al lui int, cu maxInclusive setat la 32767 și minInclusive setat la –32768.
- **byte** – Un subtip în stil IEEE al lui short, cu maxInclusive setat la 127 și minInclusive setat la –128.
- **nonNegativeInteger** – Un subset evident al lui integer, cu fațetele scale, precision și minInclusive setate la 0. Cu alte cuvinte, are utilele fațete de constrângere maxInclusive, maxExclusive, minExclusive, pattern și enumeration.
- **unsignedLong** – Un subtip în stil IEEE al lui nonNegativeInteger, cu maxInclusive setat la 18446744073709551615.
- **unsignedInt** – Un subtip în stil IEEE al lui unsignedLong, cu maxInclusive setat la 4294967295.
- **unsignedShort** – Un subtip în stil IEEE al lui unsignedInt, cu maxInclusive setat la 65535.
- **unsignedByte** – Un subtip în stil IEEE al lui unsignedShort, cu maxInclusive setat la 255.
- **positiveInteger** – Un subset evident al lui nonNegativeInteger, cu fațetele scale și precision setate la 0, iar minInclusive setat la 1. Cu alte cuvinte, are utilele fațete de constrângere maxInclusive, maxExclusive, minExclusive, pattern și enumeration.
- **nonPositiveInteger** – Un subset evident al lui integer, cu fațetele scale, precision și maxInclusive setate la 0. Cu alte cuvinte, are utilele fațete de constrângere maxExclusive, minInclusive, minExclusive, pattern și enumeration.
- **negativeInteger** – Un subset evident al lui nonPositiveInteger, cu fațetele scale și precision setate la 0, iar maxInclusive setat la -1. Cu alte cuvinte, are utilele fațete de constrângere maxExclusive, minInclusive, minExclusive, pattern și enumeration.
- **timeInstant** – Dată și oră în format ISO 8601, dar având suplimentar suportul pentru ani negativi (dar nu 0) și valori arbitrare imense pentru ani, bazate pe revizuirea propusă a ISO 8601. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, pattern și enumeration.
- **timeDuration** – Un interval de timp în format ISO 8601. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, pattern și enumeration.
- **recurringDuration** – Un interval de timp recurent în format ISO 8601. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, duration, period, pattern și enumeration.
- **recurringDate** – Un subtip al lui recurringDuration. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, duration, period, pattern și enumeration.
- **recurringDay** – Un subtip al lui recurringDuration. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, duration, period, pattern și enumeration.
- **time** – Un subtip al lui recurringDuration. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, duration, period, pattern și enumeration.
- **timePeriod** – Un subtip al lui recurringDuration. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, duration, period, pattern și enumeration.
- **date** – Un subtip al lui timePeriod. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, duration, period, pattern și enumeration.
- **month** – Un subtip al lui timePeriod. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, duration, period, pattern și enumeration.
- **year** – Un subtip al lui timePeriod. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, duration, period, pattern și enumeration.

- **century** – Un subtip al lui timePeriod. Are fațetele de constrângere maxInclusive, maxExclusive, minInclusive, minExclusive, duration, period, pattern și enumeration.
- **binary** – Dată binară. Are fațetele de constrângere length, pattern, enumeration, minlength, maxlength și encoding. encoding poate fi hex sau base64.
- **uri-reference** – Un URI. Are fațetele de constrângere length, minLength, maxLength, pattern și enumeration.
- **language** – Un cod de limbă RFC-1766. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **id** – Tipul de atribut ID din Recomandarea XML 1.0. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **idref** – Tipul de atribut IDREF din Recomandarea XML 1.0. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **idrefs** – Tipul de atribut IDREFS din Recomandarea XML 1.0. O derivare evidentă a lui idref. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **entity** – Tipul de atribut ENTITY din Recomandarea XML 1.0. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **entities** – Tipul de atribut ENTITIES din Recomandarea XML 1.0. O derivare evidentă a lui entity. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **notation** – Tipul de atribut NOTATION din Recomandarea XML 1.0. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **notations** – Tipul de atribut NOTATIONS din Recomandarea XML 1.0. O derivare evidentă a lui notation. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **qname** – Un *nume calificat* din Recomandarea XML 1.0. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **nmtoken** – Tipul de atribut NMTOKEN din Recomandarea XML 1.0. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **nmtokens** – Tipul de atribut NMTOKENS din Recomandarea XML 1.0. O derivare evidentă a lui nmtoken. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **name** – Un *nume* din Recomandarea XML 1.0. O derivare evidentă a lui entity. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.
- **ncname** – Un *nume*, fără caracterul două puncte, din Recomandarea XML 1.0. O derivare evidentă a lui name. Are fațetele de constrângere length, minLength, maxLength, pattern, enumeration, maxInclusive, maxExclusive, minInclusive și minExclusive.

Definirea şi extinderea tipurilor de date de bază

În ultima schiță a XML Schema, tipurile de date ale schemelor pot fi utilizate ca o bază de construire a altor tipuri de date, prin moștenire, într-o manieră orientată spre obiecte. Noile tipuri de date pot fi construite fie prin extinderea, fie prin restrângerea tipurilor de date existente și orice tip de dată poate fi utilizat ca prototip pentru construirea altor tipuri de date.

Definiția de element următoare apelează un prototip definit anterior, care poate conține un număr de elemente, ca un fel de facilitate macro:

```
<element name="element-name" type="ArchetypeName" />
```

Același lucru poate fi făcut și în cazul atributelor:

```
<attribute name="attribute-name" type="ArchetypeName" />
```

Definirea modelelor de conţinut

Modelul de conținut de bază al XML Schema reflectă DTD-ul XML 1.0. Cu alte cuvinte, conținutul poate fi `choice`, care este o selecție alternativă dintr-o listă, echivalentul separatorului "|"; `sequence`, care este o listă secvențială, echivalentul separatorului ","; sau `any`, care înseamnă orice, echivalentul conținutului „mixt" din XML 1.0, indicat de `(content elements)*`. În plus, XML Schema permite specificarea unei mulțimi de elemente obligatorii, dar care pot să apară în orice ordine, `all`. Cea mai obișnuită metodă de specificare a unui model de conținut este cu un element grup:

```
<complexType name="TypeName">
  <group name="groupName"
         id="ID"
         maxOccurs="number"
         minOccurs="1"
         ref="name">
<!9-- Content:  ( annotation? , ( element | group | all | choice |
                sequence | any )* ) -->
  </group>
</complexType>
```

Constrângerea conţinutului elementului şi atributului

De fapt, expresiile regulate sunt un concept dezvoltat pentru prima dată și făcut disponibil pe larg în UNIX și se referă la un limbaj de potrivire a modelelor, util pentru multe scopuri, de la căutarea anumitor tipuri de șiruri până la traducerea caracter cu caracter, sau editarea și compararea șirurilor.

Expresiile regulate pot fi folosite pentru o mulțime de constrângeri asupra conținutului atributelor și elementelor. Puteți, de exemplu, să restricționați intrările la caracterele alfanumerice ASCII și la spații, utilizând următoarea expresie regulată pentru a elimina posibilitatea de a insera într-o bază de date caractere de punctuație Unicode sau ASCII arbitrare:

```
[a-z,A-Z,Ø-9, ]*
```

În acest exemplu, virgulele sunt utilizate numai pentru a face expresia complexă mai ușor de înțeles. Ele pot fi omise fără a afecta în vreun fel înțelesul expresiei.

O expresie regulată care să limiteze introducerea unui cod poștal de cinci cifre poate arăta astfel:

```
[Ø-9][Ø-9][Ø-9][Ø-9][Ø-9]
```

În unele situații există alte metode pentru a obține aceeași limitare la intrare. De exemplu, un cod poștal de cinci cifre poate fi exprimat și ca un întreg de cinci cifre. Totuși, aceasta este mai puțin definitorie decât expresia regulată corespunzătoare, deoarece codurile poștale de cinci cifre pot avea zerouri premergătoare și trebuie să aibă o lungime de exact cinci cifre. Exprimat ca un întreg, 12 este echivalent cu 00012, dar numai 00012 este un cod poștal formatat corespunzător.

Expresiile regulate pot deveni complexe, iar explicarea utilizării lor exacte pentru a crea descrieri precise ale șirurilor țintă nu face subiectul acestui capitol. Pentru o explicație completă, vedeți anexa din specificația XML Schema Part 2: Datatypes. Pentru mai multe exemple de utilizare a expresiilor regulate, vedeți orice manual UNIX bun despre editoarele de text vi și/sau ed, sau o carte dedicată acestora, cum ar fi *Mastering Regular Expressions: Powerful Techniques for Perl and Other Tools* de Jeffrey E. F. Friedl, O'Reilly, 1997. O referință Vi bună poate fi *Learning the vi Editor, 6th Edition,* de Linda Lamb & Arnold Robbins, O'Reilly, 1998.

Validarea conţinutului elementului

În ciuda faptului că setul actual de instrumente de validare este minimal, acestea sunt foarte puternice. Puteți, de exemplu, să declarați valori minime și maxime pentru câmpuri numerice sau să utilizați expresii regulate pentru a valida formatul unui câmp de intrare de tip șir de caractere. Următoarea expresie regulată ar valida formatul unui număr de telefon valabil în America de Nord, (999) 999-9999:

```
(\(\d{3}\) )?\d{3}-\d{4}
```

Verificarea caracterului complet al înregistrărilor

În afară de aptitudinea fundamentală de a utiliza expresii regulate Unicode pentru a valida datele incluse, nu există o metodă încorporată care să asigure că un câmp este completat. Pentru numere, valorile minime și maxime pot semnala, de asemenea, un câmp vid, cel puțin din punct de vedere logic. Iată un exemplu care ar fi suficient pentru un șir de caractere și ar preveni transferarea către aplicație a unor spații albe. Evident, într-un exemplu din lumea reală poate vor trebui incluse și alte verificări ale datelor:

```
<element name="element-name"
  type="string"
  pattern="[^\t, ,\u000a,\u000d,\u0085,\u000b,\u000c,\u2029,\u2028]*" />
```

În funcție de implementare, și codul următor ar putea funcționa, dar ar trebui testat înainte. Mulți programatori cred că acolo sunt numai trei sau patru caractere spațiu alb. Observați, de asemenea, utilizarea secvenței de metacaractere `\u` în exemplul precedent pentru a semnala începutul unei valori Unicode hexazecimale:

```
<element name="element-name"
  type="string"
  pattern="[^\s]*" />
```

Fragmentele precedente ar returna o eroare dacă în câmp ar apărea numai una din grupările de caractere care definesc spațiile albe. Acestea includ spațiu, HT, CR, LF, VT,

FF, NL, LS şi PS. Alte limbaje pot avea idei diferite despre aspectul spaţiilor albe, deci este posibil să mai aveţi de adăugat la această mulţime. Cealaltă ambiguitate este tratarea caracterelor neutilizate sau a celor de control care fac parte din setul de caractere ţintă. Deoarece caracterele spaţiu alb obişnuite au fost excluse, inclusiv unele rar întâlnite, ar trebui excluse şi restul caracterelor de control? Dar seturile de simboluri care nu sunt alfabetice? Dar caracterele chinezeşti? Sunt decizii care pot însemna lucruri diferite în aplicaţii diferite; luaţi-le în considerare înainte de a trece la un anumit test.

Limitările verificării erorilor şi a validităţii

Atunci când se proiectează o bază de date, oamenii privesc informaţia conţinută deja în înregistrările existente şi încearcă să proiecteze un sistem care să captureze totul, fără redundanţe sau pierderi. Chiar şi cele mai bune realizări pot eşua uneori la conversia înregistrărilor de pe hârtie sau chiar digitale, din cauza tendinţei naturale a omului de a forţa limitele atunci când o structură existentă eşuează în capturarea nevoilor schimbătoare şi neprevăzute. Notele de pe margine pot să apară în cazul înregistrărilor pe hârtie, iar câmpurile folosite rar, sau parţial vide, pot fi folosite pentru categorii noi de informaţii. Fără analiza exhaustivă a înregistrărilor existente, ar fi dificil de garantat caracterul complet sau acurateţea.

Verificarea erorilor combină această problemă şi are potenţialul de a interzice accesul la câmpuri pe care oamenii le utilizau „neoficial" pentru a face lucruri care nu au existat niciodată în specificaţiile proiectului original. La verificarea erorilor şi limitelor într-un document XML, nu este o idee bună să depăşiţi orice verificare a erorilor sau a limitelor încorporată în aplicaţia nativă pe care documentul XML trebuie să o reflecte. Fără analizarea fiecărei înregistrări şi interogarea fiecărui utilizator, nu puteţi garanta că un câmp etichetat clar „număr casetă" nu a fost utilizat pentru stocarea de identificatori pentru locuri de parcare şi/sau pentru depozite, în cazul unor articole care nu încap în casete.

O altă problemă este creşterea nevoilor neprevăzute care afectează câmpurile existente cu un scop clar, dar limitat.

Limitarea accidentală a unei aplicaţii XML

De exemplu, plasarea de limite nefireşti sau extrem de restrictive asupra formatului informaţiei din domeniu este un lucru obişnuit la proiectanţii de baze de date, provocând astfel dificultăţi inutile pentru viitorii utilizatori ai acesteia.

În mod obişnuit, numerele de telefon din SUA au o lungime de şapte sau unsprezece cifre, după cum sunt locale sau interurbane. În trecut, formatul prefixelor interzicea numere cu un 1 sau cu un 0 în poziţia centrală.

Foarte multe câmpuri de introducere a datelor au fost deja create, ceea ce implică faptul că aceste numere trebuie să fie introduse într-un format exact, cum ar fi 1 (909) 999-9999 sau 999-9999 cu un set restrictiv de limitări referitoare la prefixele tradiţionale.

O dată cu formarea obligatorie a celor unsprezece cifre în multe localităţi şi cu modificările în formatul prefixului, care admite acum orice cifră în poziţia centrală, a fost necesar ca multe dintre aceste scheme să fie înlocuite, cu preţul multor ore de efort de programare.

În plus, lumea este un loc mai mare decât ne-ar putea face să credem experienţa noastră limitată la Canada şi Statele Unite. Corporaţiile multinaţionale pot avea un client în Bordeaux, Franţa, al cărui număr poate fi +33 (0)556 99 99 99 99, cu 14 cifre în loc de 11, care se întâmplă să corespundă unui standard internaţional pentru formatul de prezentare a numerelor de telefon, care utilizează paranteze în cu totul alt scop decât în Statele Unite.

Pe plan internaţional, numărul de cifre al numerelor de telefon locale poate diferi chiar şi în cadrul aceluiaşi oraş. Codurile oraşelor, pe care le numim prefixe, pot diferi şi ele în lungime. Codul de ţară diferă de la o ţară la alta. Codul „1" al SUA contrastează cu „44" al Angliei, cu „353" al Irlandei, iar posibilitatea apariţiei codurilor de ţări de patru cifre creşte de la an la an, o dată cu apariţia de noi state independente. Armata SUA şi alte servicii militare utilizează linii de comunicaţii private, care pot folosi, pe lângă cifrele de la 0 la 9, şi „numere" hexazecimale „A", „B", „C" şi „D".

Naivele scheme de validare a numerelor de telefon din mii de aplicaţii, care se confruntă acum cu o comunitate internaţională sau militară, trebuie actualizate, la un cost imens, înainte de a se putea adresa acelor piete. Dacă a fost utilizată o validare restrictivă a prefixului, ea trebuie eliminată pentru a putea manevra pieţele în extindere din SUA şi Canada. Înainte de construirea unui câmp pentru numere de telefon şi de validarea cifrelor, asiguraţi-vă că ştiţi într-adevăr cu ce fel de date poate fi completat pe termen lung şi faceţi clar cunoscute aşteptările dumneavoastră sub forma unor instrucţiuni pe ecran sau a unui şablon. Gândiţi-vă să nu validaţi deloc.

În acelaşi mod, poziţia şi forma titlurilor, codurilor poştale, metodelor de adresare poştală, onorurilor, gradelor, numerelor şi aşa mai departe diferă de la o ţară la alta.

Utilizarea de prea multe ori a unor scheme restrictive impuse lumii de către programatorii a căror idee de universalitate este limitată la permiterea de hamburgeri cu sau fără ceapă este o metodă garantată de a-i îndepărta şi supăra pe clienţii potenţiali.

Câmpuri de intrare cu format liber

În multe situaţii, singura metodă de a cuprinde toată lumea este de a permite câmpuri de intrare cu format liber, aşa cum numele de străzi pot fi şiruri arbitrare. 100 Main Street în Pocatello din Idaho poate fi 100 Avenida de las Ramblas în San Clemente din California, sau 100 East 500 North în Salt Lake City din Utah. În unele zone din Japonia, o adresă poate arăta astfel: Shokado, Nishi-iru, Nishitoin, Shimodachiura, Kamigyo-ku, Kyoto, Japan. Care este numele străzii în exemplul japonez? Există oare un cod poştal? Cum comprimaţi această înregistrare într-un set de câmpuri Adresă, Oraş, Stat, Cod poştal? Înregistrările bazei dumneavoastră de date permit oare introducerea unui nume de ţară?

Codurile poştale pot fi plasate înainte sau după numele oraşului şi poate chiar înainte de numele străzii. Într-o lume a formatului liber, şi bazele noastre de date trebuie să aibă într-o oarecare măsură un format liber. Cu toate că XML 1.0 în sine permite ca datele să fie definite ca şi când ar fi prezente în orice ordine, aceleaşi câmpuri pot cere o anumită ordine în prezentarea lor, care poate diferi de la o ţară la alta. Asiguraţi-vă că orice bază de date subiacentă sau transformare a bazei de date nu amestecă sau nu distruge informaţia de ordonare, ca nu cumva să nu puteţi să o refaceţi. Acestea nu sunt lucruri lipsite de importanţă. Adresele formatate corespunzător pot pregăti compania dumneavoastră pentru reduceri substanţiale ale costurilor expedierii prin poştă. Adresele incorecte pot trimite scrisorile şi coletele dumneavoastră începişor înspre abis.

Propunerile anterioare de scheme XML

Au fost o serie de propuneri către W3C încercând să rezolve părți ale problemei de verificare a erorilor și a validității.

Precursoarea tuturor propunerilor de scheme este forma arhitecturală Hy-Time, care este o parte a lumii SGML. Asemenea majorității SGML și în special în cazul Hy-Time, formele arhitecturale au o putere imensă. Formele arhitecturale sunt o colecție de reguli pentru extinderea și modificarea elementelor după dorință. În loc să ajustați DTD-ul cu meta-descrieri încapsulând elementele XML în sine, așa cum fac următoarele trei propuneri, formele arhitecturale vă permit să modificați structura reală a elementelor existente, realizând de fapt orice prelucrare a elementului pe care v-o puteți imagina, fără a modifica deloc comportamentul DTD-ului. Din păcate, formele arhitecturale sunt înzestrate în același timp, ca și Lamont Cranston (Umbra – The Shadow), și cu puterea de a umbri mințile oamenilor. Implementarea peste XML, care se presupune că este simplu și ușor de utilizat, a unui limbaj de prelucrare extraordinar de complex este improbabil să se realizeze în viitorul apropiat (sau chiar îndepărtat). Este contrară întregii filozofii din spatele XML, simplitatea, și va rămâne fără îndoială pentru totdeauna în lumea Hy-Time.

XML-Data

Propunerea Microsoft a fost XML-Data, care are facilități excelente pentru date, un mecanism de zone de nume, pentru unii mult prea legat de o facilitate de rezoluție de tip Windows Registry, și multe alte caracteristici iubite de programatori. Este cât se poate de adaptat la genul de lucruri făcute de programatorii Windows și mai puțin prietenos față de alte medii, deci, ca de obicei, au apărut unele proteste. O slăbiciune importantă este că utilizează o sintaxă unică pentru a descrie schema și trebuie comentată, asemenea instrucțiunilor CSS, pentru a evita redarea necorespunzătoare. Deoarece Recomandarea XML 1.0 face opțional transferul de comentarii către aplicații și descurajează chiar ideea utilizării comentariilor pentru cod, acesta pare să fie un defect fatal.

Resource Description Framework (RDF – cadrul descrierii resurselor)

Contribuția Netscape a fost Resource Description Framework, RDF, care este o generalizare a altor instrumente de descriere a resurselor cum ar fi Platform for Internet Content Selection, PICS, și Dublin Core. Nu adresează de fapt același spațiu de date și nici măcar aceleași probleme, dar a fost văzut ca un rival pentru propunerea Microsoft. Asemenea propunerii Microsoft, RDF utilizează o zonă de nume XML pentru a identifica descrierile metadată ale articolelor de date dintr-un DTD XML. Spre deosebire de XML-Data, sintaxa este cea de declarare obișnuită a elementelor și atributelor XML, deci în XML nu trebuie modificat nimic pentru a face ca totul să funcționeze. Dezvoltarea RDF continuă, dar nu a fost și nu este proiectat pentru a adresa întregul spațiu al problemei schemelor.

IBM a încercat să pună la un loc părți din fiecare propunere, sub forma Document Content Description, DCD. Asemenea ambelor propuneri anterioare, Microsoft și Netscape, aceasta utilizează DTD-ul XML și îl extinde cu declarații speciale. DCD combină unele dintre cele mai bune caracteristici ale fiecărei propuneri explicate anterior. Ca și XML-Data, DCD are facilități puternice pentru tipurile de date. Ca și RDF, este foarte flexibil și nu trece prea mult în afara structurii DTD-ului XML pentru a permite descrieri complete ale elementelor XML.

XSchema

Lista poștală XML-DEV a pus în circulație propria propunere, XSchema, care a evoluat în Document Definition Markup Language, DDML. DDML este mult mai orientat spre obiecte și, în loc să utilizeze declarații de tip DTD pentru a codifica metadate, pentru a descrie schema el utilizează un document structurat XML obișnuit. Toate constrângerile asupra datelor documentului sunt codate utilizând imbricarea elementelor și atributelor. DDML este puțin mai ambițios decât oricare dintre cele trei propuneri prezentate anterior, prin faptul că încearcă să înlocuiască complet DTD-ul XML. El propune instrumente pentru conversia automată a DTD-urilor XML în documente DDML și viceversa. Deși ar trebui să apară și ar avea în mod sigur avantaje, mi se pare că XML și XML Schema au un avans mult prea mare și un enorm angajament din partea unui număr mare de furnizori, deci în contextul acestei cărți probabil este un candidat fără șanse.

Schema for Object-Oriented XML (SOX)

În încheiere, SOX, Schema for Object-Oriented XML (care ar trebui să fie de fapt SOOX), încearcă să extindă XML în regatul orientării complete spre obiecte. Sunt convinsă că elementele SOX și chiar DDML vor fi încorporate în XML Schema. Dar W3C a lăsat întotdeauna ștacheta destul de jos atunci când a recomandat standarde, pentru a nu lăsa de izbelițe prea multe companii. Majoritatea propunerilor care trec în stadiul de recomandare sunt atât practice cât și realizabile într-o perioadă de timp rezonabilă, astfel încât avantajele competitive relative ale companiilor care propun standardele sunt minimizate.

Depanarea schemelor XML

Setarea valorilor prestabilite

Doriți să furnizați valori prestabilite pentru anumite câmpuri, fără a modifica DTD-ul documentului. Acest lucru poate fi făcut cu ușurință pentru atribute și, deoarece XML Schema reproduce întocmai atât de multe caracteristici ale DTD-ului, acest lucru este puțin probabil să se modifice. Gândirea actuală a W3C pare să fie înclinată spre permiterea valorilor prestabilite și în conținutul elementelor, dar aceasta nu este încă bătută în cuie. Pentru setarea unei valori prestabilite sau a uneia fixate pentru un atribut, utilizați formatul următor:

```
<attribute name="attribute-name"
  default="value" | fixed="value"
  type="string" />
```

Pentru a seta același gen de valori pentru un element, este utilizat același format general, dar poate fi setat atât pentru prototipul care definește elementul, cât și pentru elementul în sine:

```
<element name="element-name"
  default="value" | fixed="value"
  type="string" />
```

sau

```
<archetype name="ArchetypeName"
  default="value" | fixed="value"
  type="string" />
```

Ambele mecanisme se pot modifica, dar acestea și toate celelalte exemple vor fi publicate într-o formă actualizată pe situl meu Web, la `http://www.leeanne.com/usingxml/`, în afară de cazul în care această aptitudine nu va dispărea complet, fapt care va fi de asemenea anunțat.

XPath

CAPITOLUL

8

În acest capitol

- Identificarea cu XPath a locațiilor din document
- Urmărirea poziției dumneavoastră într-un document
- Utilizarea abrevierilor XPath
- Funcții principale XPath
- Utilizarea predicatelor XPath
- Probleme cu ancorele și cu legăturile HTML
- Studii de caz

Identificarea cu XPath a locaţiilor din document

XPath este o tehnologie care vă permite să identificați părți ale unui document XML. Această funcționalitate este partajată între XSL, XSLT, XPointer și alte standarde care necesită abilitatea de a localiza poziții în documente. Noțiunea de adresare a părților unui document printr-un limbaj clar, dar fără a vă îngropa în complexitatea modelului document obiect (DOM – Document Object Model), este mult prea bună pentru a o păstra numai pentru dumneavoastră. După cum indică și standardele în sine, pointerii suplimentari și limbajele de identificare a fragmentelor pot fi definite eventual pentru SMIL sau SVG, printre mulți alți candidați, ca o mare parte a întregului curent al dezvoltării standardelor actuale. W3C modularizează și extinde standardele existente pentru a le face utilizabile în cât mai multe contexte cu putință, cu cheltuieli minime.

XPath este un limbaj special care vă permite să descrieți locația unui anumit punct dintr-un document XML printr-o parcurgere din aproape în aproape a arborelui documentului, utilizând ca punct de plecare orice nod. După cum s-a menționat anterior, el formează terenul comun între XSL, XSL Transformations, XLink (XLL) și XPointer. Împreună, ele permit mecanisme flexibile și extensibile pentru legarea documentelor structurate și pentru multe alte aplicații. Mecanismele pointer anterioare cereau ca elementele ancoră fizice să fie plasate în fișierele la care se leagă, dacă era nevoie de altceva decât de acces la nivelul superior. Nevoile actuale fac acest lucru imposibil, deoarece numărul de metode prin care alții pot dori să facă legături la un document este necunoscut, iar documentul în sine se poate modifica în mod dinamic.

Deoarece XPath definește o structură canonică pentru document astfel încât orice legătură externă sau internă să o poată accesa pentru a identifica noduri particulare în

arborele unui document, și este o parte atât din XLink cât și din XSL, XPath este prezentat în acest capitol, iar XLink și XPointer sunt prezentate în Capitolul 9, „XLink și XPointer".

Ca și celelalte standarde înrudite cu XML, XPath este proiectat să funcționeze cu alte standarde W3C, deci ar trebui să fie la fel de comodă indicarea părților unui document HTML (construit corect), RDF sau ale oricărui document tip XML construit corect.

Un XPath are trei părți principale: o direcție de traversare numită *axă*, un test pentru nod care alege noduri de pe acea linie de traversare și un predicat algoritmic care poate fi utilizat pentru o rafinare suplimentară a nodului selectat. Vom cerceta amănunțit fiecare parte în restul acestui capitol.

Axele XPath

Deoarece puteți porni un XPath din orice nod, trebuie ales un cuvânt care să descrie direcția parcurgerii. Din păcate, cuvântul ales nu este nici inteligibil în sine, și nici elocvent în privința faptului că reprezintă o direcție relativă într-un arbore ordonat.

Axe este unul dintre acele cuvinte extrem de tehnice care specifică cuvintele cheie direcționale care vă permit să navigați la diverse locuri dintr-un document bazându-vă pe locația dumneavoastră actuală. Seamănă mai mult cu indicații de conducere decât cu o axă de coordonate sau cu axul unei roți. În ciuda faptului că oricine poate fi impresionat de oamenii care au studiat matematica elementului finit și logica predicatelor, informaticienii ar trebui să se ferească de nevoia de a impresiona pe alții cu extraordinara precizie științifică a acestor cuvinte, altfel publicul larg îi va considera niște tocilari nefericiți.

XPath utilizează perspectiva metaforică a unei furnici care se plimbă pe un arbore, și nu o schemă arbitrară de localizare. Dumneavoastră, programatorul, sunteți locatarul temporar al fiecărui nod prin care treceți, traversând arborele din aproape în aproape. Deci navigați folosindu-vă de jaloane și de relații între noduri, așa cum fac mulți și în viața reală. Simplitatea și economia acestei scheme, spre deosebire de localizarea nodurilor prin coordonatele x, y, după cum ar sugera cuvântul *axă*, pare să se potrivească mai mult cu un cuvânt cald cum este „relație" decât cu „axe", un cuvânt atât de tehnic. Deci gândiți-vă la relații XPath și pretindeți că *axe* este un cuvânt științific.

Iată câteva chestiuni pe care trebuie să le rețineți pentru a înțelege relațiile (axele) XPath:

- XPath tratează documentul XML ca pe un arbore de noduri, unele care au nume (elementele, atributele și zonele de nume) și unele care nu au (valorile atributelor și textul). Numai elementele participă complet în relații. Nici atributele și nici textul nu pot avea noduri descendente, iar atributele, de obicei, nu se întâlnesc ca noduri.
- Deoarece nodurile element sunt singurele lucruri cu nume, XPath utilizează întotdeauna ca bază pentru descrierile locației numele unui anumit nod, în specificație numit *context*. O excepție minoră există în nodul rădăcină, care are numele „/" și nu altul. Nodul rădăcină, sau orice nod cu nume, poate fi utilizat ca o bază pentru o cale relativă, modificând prin aceasta locația nodului context. Dar cu toate că numele nodului curent, sau context, poate să nu fie cunoscut, acel nume este disponibil întotdeauna prin utilizarea funcțiilor `local-name()` și `namespace-uri()`.
- XPath suportă zonele de nume XML, deci un nume de nod poate fi calificat de o zonă de nume, care este chiar ea o axă.
- XPath suportă două tipuri de căi de locație, căi de locație relative și căi de locație absolute. Căile de locație relative pornesc prin relație de la un nod context, altul decât rădăcina. Căile de locație absolute pornesc din rădăcină. Primul exemplu de cale este o cale relativă, iar cel de-al doilea este o cale absolută:

```
child::division/child::paragraph
/child::book/child::div/child::para
```

Primul exemplu selectează toate paragrafele copil ale copilului `division`, oricare ar fi nodul curent sau nodul context. Pe de altă parte, cel de-al doilea exemplu selectează aceleași elemente, dar numai dacă sunt descendenți direcți ai unui document `book`.

- Deoarece sintaxa XPath completă folosește atât de multe cuvinte, există multe abrevieri pentru a simplifica descrierile locației și a le face mai ușor de înțeles. Pentru locațiile exacte din exemplul de mai sus, sintaxa abreviată ar fi:

```
division/paragraph
/book/division/paragraph
```

Cu toate că aceasta nu pare să facă prea multă economie, atunci când expresia locației devine complicată, scăpați de un număr semnificativ de acționări de taste. În secțiunea „Abrevieri" aveți o listă de prescurtări.

- XPath vă permite, de asemenea, să selectați elementele după conținutul atributelor lor, deci în sintaxa abreviată

```
division/paragraph[@type="firstParagraph"]
```

selectează toate paragrafele descendente ale descendentului `division`, oricare ar fi nodul curent sau nodul context, dar numai dacă ele conțin un atribut numit `type` și având valoarea `"firstParagraph"`. De exemplu, atunci când utilizați XPath în combinație cu XSLT, această posibilitate vă permite să tratați primele paragrafe altfel decât pe celelalte, în cazul în care doriți să inserați letrine sau să aplicați alt tratament special pentru text.

Lista de mai sus descrie axele XPath pe scurt, cu toate că modurile în care pot fi combinate sau în care pot interacționa sunt aproape infinite.

Nodurile XPath

În secțiunea precedentă, cuvântul *nod* a fost folosit destul de frecvent. Dar *nod* este o altă alegere nefericită de nume din partea grupului de lucru și oglindește o încercare de a extinde terminologia grafurilor dincolo de capacitatea de înțelegere a muritorilor de rând, care nu au studii matematice.

Nodurile atribut există într-un fel de a treia dimensiune și nu fac parte efectiv din arborele bidimensional al documentului. Ele sunt definite, de fapt, ca fiind fără părinți și fără copii, în ciuda faptului că există o paralelă clară între conținutul (valoarea) unui atribut și conținutul text al unui element.

Singura modalitate de a înțelege atributele este să vă gândiți că acestea există în interiorul elementelor – care sunt noduri reale, cu descendenți și strămoși. Atributele, pe de altă parte, sunt într-un fel nodul efectiv și au o relație mult mai intimă cu acesta decât o descendență simplă. Ele sunt create o dată cu nodul și mor o dată cu el. Vă puteți gândi la ele mai mult ca la organe interne decât ca la copii. Inima și plămânii dumneavoastră nu sunt copiii dumneavoastră, în ciuda faptului că există o legătură evidentă și intimă. Nici dumneavoastră nu sunteți părintele lor, deoarece ele iau ființă o dată cu dumneavoastră, provenind din ADN-ul dumneavoastră. Deci utilizarea cuvântului *nod* pentru a le descrie creează numai confuzie și haos.

De asemenea, nodurile text au o relație specială cu nodul lor părinte. În primul rând, ele sunt sterile și nu pot avea nici descendenți și nici atribute. Aceasta este o diferență importantă, deci etichetarea lor cu același nume nu prea are sens, indiferent de modul în care satisface prejudecățile logice ale teoriei grafurilor. Traversarea unei secțiuni mari de text, asemenea unei serii ordonate de semi-noduri frați, are sens, deoarece sunt copiii părinților lor imediați, devenind astfel parte componentă a arborelui documentului.

Totuși, ele se comportă mult diferit față de nodurile document adevărate, deoarece analizorului XML însuși îi este permis să le creeze și să le distrugă, interceptând de exemplu secvențe de salturi la rând nou și înlocuindu-le cu retururi de car sau cu un singur spațiu, sau înlocuind secvențe de caractere Unicode cu echivalentele lor *normalizate*. Un analizor XML nu are nici o șansă să facă așa ceva cu un nod adevărat.

Această dualitate este de la sine înțeleasă în text, deoarece este de fapt atât un șir, cât și o serie ordonată de noduri, dar el balansează între caracterul său de nod și cel de șir. Discuția despre care dintre aceste caractere este „real" este o problemă care seamănă mai mult cu mecanica cuantică, poate chiar cu metafizica, decât cu matematica elementului finit.

Într-un model de conținut combinat, textul poate avea elemente efective înglobate în el, ca și stafidele într-o budincă, dar aceste elemente sunt copiii părintelui, și nu copiii textului. De fapt, textul continuă prin elementul încadrat ca un electron care străpunge o barieră cuantică de potențial sau ca nava spațială Enterprise prin hiperspațiu. Fără acest comportament, compararea șirurilor este aproape imposibilă, deoarece „acest *șir*" (scris cursiv) nu va fi echivalent cu „acest **șir**" (scris cu aldine). În practică, s-ar putea să fie nevoie să forțați acest comportament, în DOM, cu un indicator `whatToShow`. Pentru detalii vezi Capitolul 10, „DOM (Document Object Model) – Modelul document obiect".

▶ Pentru mai multe informații referitoare la vederea logică, **vezi** „Controlul vederii logice a arborelui documentului", la **pagina 213.**

Urmărirea poziției dumneavoastră într-un document

Expresia *nod context* va fi utilizată în acest capitol numai datorită faptului că așa îi spune specificația, deci ar trebui să vă obișnuiți să recunoașteți termenul. Acesta înseamnă de fapt nodul curent, cel cu care lucrați la un moment dat. Echivalentul într-un document text este locația cursorului.

Lista următoare descrie relațiile fundamentale dintre noduri, în termenii nodului curent sau nodului context. Părți ale descrierilor următoare sunt luate sau extinse din Recomandarea XPath a W3C, care poate fi vizualizată în totalitate la adresa `http://www.w3.org/TR/`:

- **child (copil)** – Această relație conține copiii nodului curent sau context, prin care se înțelege locația actuală a nodului „cursor".
- **descendent** – Această relație conține descendenții nodului curent sau context. Un descendent este un copil, sau un copil al unui copil și așa mai departe. Axa descendentului nu conține niciodată noduri atribut sau zonă de nume, dintr-un motiv explicat mai târziu, la punctele referitoare la frați.
- **parent (părinte)** – Această relație conține părintele nodului curent sau context. Cu toate că specificația actuală implică faptul că părintele unui atribut este nodul care îl conține, aceasta ridică o problemă importantă de asimetrie: cum poate un atribut să aibă un nod ca părinte, dacă nodul nu are un copil care să fie un atribut? Atributele nu participă de fapt în relația părinte, copil sau frate în acest mod și sunt mai mult ca și calitățile personale la oameni. Așa cum inteligența sau entuziasmul dumneavoastră nu pot fi descrise decât metaforic ca având o legătură între ele, și cum nici mama dumneavoastră nu poate fi părintele cinstei dumneavoastră decât (iarăși) metaforic, atributele nu au părinți sau frați *reali*.

- **ancestor (strămoș)** – Această relație conține strămoșii nodului curent sau context. Un strămoș este un părinte, sau părintele părintelui și așa mai departe. Sunt valabile aceleași precizări ca și pentru nodurile atribut sau zonă de nume.
- **following-sibling (următorul frate)** – Această relație conține toți frații următori ai nodului curent sau context. Dacă nodul curent sau context este un nod atribut sau un nod zonă de nume, axa următorul frate este vidă. Aceasta este o modalitate pretențioasă de a spune că dintr-o frunză atribut nu vă puteți deplasa oriunde, decât dacă prima dată vă întoarceți la nodul părinte care o conține. Deoarece atributele pot fi adăugate oricând la un element prin modificarea DTD-ului fie în submulțimea externă a DTD-ului fie în cea internă, nu vă puteți baza pe ele pentru o parcurgere completă.
- **preceding-sibling (fratele precedent)** – Această relație conține toți frații precedenți ai nodului curent sau context. După cum s-a menționat anterior, dacă nodul context este un nod atribut sau un nod zonă de nume, axa fratele precedent este vidă.
- **following (următorul)** – Această relație conține toate nodurile din același document cu nodul curent sau context care se află după nodul context în ordinea documentului, excluzând toți descendenții, nodurile atribut și nodurile zonă de nume.
- **preceding (precedentul)** – Această relație conține toate nodurile din același document cu nodul curent sau context care se află înaintea nodului context în ordinea documentului, excluzând toți strămoșii, nodurile atribut și nodurile zonă de nume.
- **attribute (atribut)** – Această relație conține atributele nodului curent sau context. Axa este vidă dacă nodul context nu este un element. Cu toate că atributele participă în reprezentarea uzuală a grafului direcționat care este arborele documentului, arcul dintre un nod și atributele sale seamănă mai mult cu o linie punctată, formând mai degrabă o *aură* metafizică decât o descendență reală. Dacă vă gândiți la ele ca fiind situate într-un plan mai înalt decât relațiile fizice existente între un nod și părinții, copiii și frații săi, nu greșiți prea mult.
- **namespace (zonă de nume)** – Această relație conține nodurile zonă de nume ale nodului curent sau context. Axa este vidă dacă nodul context nu este un element. Deoarece, așa cum s-a menționat în Capitolul 6, „Zone de nume XML", o zonă de nume este un fel de prescurtare a unui atribut `xmlns`, acest lucru ar trebui să fie foarte clar acum. Și, deoarece zona de nume este de fapt un atribut, fie direct, fie ca prescurtare a prefixului zonei de nume pentru același nume, sunt valabile aceleași precizări ca și în cazul celorlalte atribute. Aceasta dă naștere unui alt lucru interesant referitor la prefixele zonă de nume ale atributelor. Deoarece prefixul unei zone de nume este echivalentul exact al unui atribut `xmlns` atunci când este utilizat pentru numele unui element, ce înseamnă de fapt atunci când atașați unui atribut un prefix de zonă de nume? Aceasta servește la sublinierea naturii metafizice a zonelor de nume și a atributelor în general.
- **self (el însuși)** – Această relație conține numai nodul curent sau context. Furnica în sine.

- **descendent-or-self (descendent sau el însuși)** – Această relație conține nodul curent sau context și toți descendenții săi. Aceasta permite proiectantului să adreseze conținutul unui element fără să îi pese dacă există subelemente sau dacă vor fi adăugate ulterior.
- **ancestor-or-self (strămoș sau el însuși)** – Această relație conține nodul curent sau context și toți strămoșii săi. Aceasta permite proiectantului să adreseze întreaga ramură a unui document în care se află elementul, indiferent dacă există niveluri adiționale de elemente strămoș sau dacă vor fi adăugate ulterior.

Observați că între ele, `ancestor`, `descendent`, `following`, `preceding` și `self` iau în considerare întregul spațiu accesibil al documentului, centrat în jurul unui singur element. Cu alte cuvinte, un document este format cu exactitate dintr-un singur element și toți strămoșii și descendenții săi, plus toate nodurile precedente, excluzând strămoșii, și toate nodurile următoare, excluzând descendenții. Această analiză exclude „nodurile" atribut și „nodurile" zonă de nume, toate fiind într-o relație metafizică cu documentul în sine, fiind astfel dificil de explicat în termenii relațiilor umane.

Toate aceste relații pot fi exprimate din punct de vedere teoretic ca și relații copil ordonate ale rădăcinii, dar, din motive de comoditate, sunt denumite explicit, relativ la orice locație. Aceasta permite nu numai o economie a exprimării, dar și o anume robustețe dacă documentul se modifică într-un mod care afectează relația diverselor noduri.

Notă Relațiile copil numerotate secvențial sunt, de fapt, baza notației de adresare în salturi, sau din aproape în aproape, descrisă în secțiunea „XPointer" din Capitolul 9.

Utilizarea abrevierilor XPath

Deoarece se presupune că locațiile XPath se utilizează atât manual cât și prin generare automată, există multe abrevieri menite să economisească acționări de taste și să mărească lizibilitatea orientării locațiilor complexe.

Acest exemplu scurt arată cum pot fi folosite abrevierile pentru a tasta mai puțin. În fragmentul de document următor, simulați că ați navigat prin document și că acum vă aflați la nodul numit „`chapter`", care astfel devine nodul context. O locație are următoarele trei părți:

1. O relație sau o axă, urmată de două caractere două puncte consecutive.
2. Un nume de nod sau un test, care poate conține informație zonă de nume. Un nume de nod sau un test poate fi urmat de o relație sau o axă suplimentară și de un al doilea nume de nod sau test, și așa mai departe.
3. Un predicat, adică o expresie care poate fi evaluată pentru a oferi informații suplimentare despre locația respectivă. Predicatele pot conține funcții și relații logice booleene. Un predicat poate conține, de asemenea, o relație sau o axă pentru a determina cum va fi aplicată expresia.

Singura parte obligatorie este numele nodului sau testul, deoarece celelalte părți au valori prestabilite. Următorul fragment de cod este utilizat în exemple:

```
<chapter>
  <title>Now and Again</title>
```



```
  <paragraph lang="en-UK">
    <quote>
      This is the colour of darkness, that civilisation not be
      obscured by grey detail.
    </quote>
  </paragraph>
  <paragraph>
    The above quote is from Prufrock, 1947...
  </paragraph>
</chapter>
```

În fragmentul de document precedent puteți selecta astfel toate elementele copil ale lui `chapter`, presupunând că `chapter` este nodul curent:

```
child::*
```

Dar deoarece relația copil este relația prestabilită în XPath, același rezultat poate fi obținut cu

```
*
```

Dacă doreați să selectați primul element `paragraph` conținut în `chapter`, puteați scrie:

```
child::paragraph[position()= 1]
```

care conține toate cele trei părți ale unei locații XPath, dar iarăși, se pot economisi câteva acționări de taste prin utilizarea unei abrevieri

```
paragraph[1]
```

lucru permis de regulile de abreviere. `child` (copil) este relația prestabilită, iar `position()` este expresia prestabilită într-un predicat.

Există multe alte abrevieri uzuale, iar permutările posibile sunt foarte multe, dar maniera în care puteți selecta un nod prin valoarea unui atribut cere o atenție specială. Sintaxa completă pentru selectarea unui element `paragraph` bazându-vă pe valoarea atributului `lang` inclus, este

```
child::paragraph[attribute::lang="en-UK"]
```

dar forma abreviată va fi

```
paragraph[@lang="en-UK"]
```

Aceasta face ca relația să fie un pic mai ușor de înțeles, fără a fi nevoit să descifrați cui îi sunt atașate caracterele două puncte.

În lista următoare, numele cursive sunt arbitrare și pot fi înlocuite cu orice nume valid. Prima este sintaxa neabreviată, apoi apare un semn egal urmat de sintaxa abreviată, dacă există, sau de N/A în caz contrar, o linie de pauză și apoi o explicație. Combinațiile posibile de abrevieri și sintaxe neabreviate sunt infinite din punct de vedere teoretic, deci această listă este întrucâtva prescurtată:

`child::`*`element-name`* **`=`** ***`element-name`*** – selectează elementul copil *element-name* al nodului curent sau context.

`child::* = *` – selectează toate elementele copil ale nodului curent sau context.

`child::text() = text()` – selectează toate nodurile text copil ale nodului curent sau context.

`attribute::`*`name`* **`=`** ***`@name`*** – selectează atributul *name* al nodului curent sau context.

`attribute::* = @*` – selectează toate atributele nodului curent sau context.

`child::element-name[position()=1] = element-name[1]` – selectează primul copil *element-name* al nodului curent sau context.

`child::element-name[position()=last()] = element-name[last()]` – selectează ultimul copil *element-name* al nodului curent sau context.

`child::*/child::element-name = */element-name` – selectează toți nepoții *element-name* ai nodului curent sau context.

`/child::document/child::chapter[position()=5]/child::section[position()=2]=/document/chapter[5]/section[2]` – selectează cea de-a doua *section* a celui de-al cincilea *chapter* din *document*, care stă în rădăcină și, evident, numele arbitrare pot fi înlocuite cu orice.

`child::chapter/descendent::element-name = chapter//element-name` – selectează toate elementele descendent *element-name* ale elementului copil *chapter* al nodului curent sau context.

`/descendent-or-self::node()/child::element-name = //element-name` – selectează toți descendenții *element-name* ai rădăcinii documentului și, astfel, selectează toate elementele *element-name* aflate în același document cu nodul curent sau context.

`/descendent-or-self::node()/child::list-name/child::item = //list-name/item` – selectează toate elementele *item* aflate în același document cu nodul curent sau context care au ca părinte un *list-name*.

`self::node() = .` – selectează nodul curent sau context.

`self::element-name – N/A` – selectează nodul curent sau context numai dacă este un *element-name*.

`/ = /` – selectează rădăcina documentului.

`self::node()/descendent-or-self::node()/child::chapter[position()=8] = .//chapter[8]` – selectează cel de-al optulea element *chapter*, descendent al nodului curent.

`/descendent-or-self::node()/child::element-name = .//element-name` – selectează elementele *element-name*, descendenți ai nodului curent sau context.

`parent::node() = ..` – selectează părintele nodului curent sau context.

`ancestor::* = N/A` – selectează toți strămoșii nodului curent sau context.

`parent::node()[attribute::language] = ../@language` – selectează atributul *language* al părintelui nodului curent sau context.

`child::element-name[attribute::language="en-UK"] = element-name[@language="en-UK"]` – selectează toți copiii *element-name* ai nodului curent sau context al căror atribut *language* are valoarea „*en-UK*".

`child::element-name[attribute::language="en-UK"][position()=5]=element-name[@language="en-UK"][5]` – selectează cel de-al cincilea copil *element-name* al nodului curent sau context al cărui atribut *language* are valoarea „*en-UK*".

`child::element-name[position()=5][attribute::language="en-UK"]=element-name[5][@language="en-UK"]` – selectează cel de-al cincilea copil *element-name* al nodului curent sau context al cărui atribut *language* are valoarea „*en-UK*".

`child::chapter[child::title="Introduction"]=chapter[title="Introduction"]` – selectează acel copil *chapter* al nodului curent sau context care are unul sau mai mulți copii *title* având valoarea șir „*Introduction*".

`child::chapter[attribute::title] = chapter[@title]` – selectează acel copil *chapter* al nodului curent sau context care are un atribut *title*.

`child::chapter[attribute::title and attribute::subtitle] = chapter[@title and @subtitle]` – selectează toți acei copii *chapter* ai nodului curent sau context care au atât un atribut *title*, cât și un atribut *subtitle*.

`child::chapter[attribute::title or attribute::subtitle]=chapter[@title or @subtitle]` – selectează toți acei copii *chapter* al nodului curent sau context care au fie un atribut *title*, fie un atribut *subtitle*.

Funcții principale XPath

Fiecare funcție din biblioteca de funcții XPath este specificată utilizând un prototip de funcție care oferă tipul returnat, numele funcției și tipul argumentelor. Dacă tipul unui argument este urmat de un semn de întrebare, atunci argumentul este opțional; altfel, argumentul este obligatoriu.

Funcții pentru noduri

Informația returnată de funcțiile pentru noduri este legată într-un fel de noduri individuale sau de mulțimi de noduri:

- *number* `last()` – Funcția `last()` returnează un număr egal cu dimensiunea contextului din contextul evaluării expresiei. Acesta este un mod obscur, dar precis, de a spune că dacă nodul curent sau context conține un șir, atunci `last()` este echivalentul aproximativ al lui `sizeof(string)`. Dar dacă poziția contextului este deplasată în interiorul nodului prin intermediul unui predicat de la locația elementului care îl conține, atunci `last()` este mai degrabă `sizeof(substring)`.
- *number* `position()` – Funcția `position()` returnează un număr egal cu poziția contextului din contextul evaluării expresiei. Acesta este un mod obscur, dar precis, de a spune că dacă nodul curent sau context conține un șir, atunci `position()` este echivalentul aproximativ al poziției cursorului caracter în acel șir. Această poziție poate fi deplasată prin intermediul unui predicat din locația nodului curent sau context.
- *number* `count(node-set)` – Funcția `count()` returnează numărul de noduri din argumentul `node-set`. Explicația este destul de clară. Dacă ați selectat un grup de noduri prin orice mecanism de selecție a nodurilor, această funcție vă spune câte noduri au fost selectate.
- *node-set* `id(object)` – Funcția `id()` selectează elementele prin ID-ul lor unic și returnează o listă `node-set`. Aceasta este o altă metodă de selectare a unei mulțimi de noduri.
- *string* `local-name(node-set?)` – Funcția `local-name()` returnează partea locală a numelui extins al nodului din argumentul `node-set`, care este primul în ordinea documentului. Dacă argumentul lipsește, atunci `node-set` este nodul curent sau context.
- *string* `namespace-uri(node-set?)` – Funcția `namespace-uri()` returnează URI-ul zonei de nume a numelui extins al primului nod în ordinea documentului, din argumentul `node-set`. Dacă argumentul lipsește, atunci `node-set` este nodul curent sau context.
- *string* `name(node-set?)` – Funcția `name()` returnează un șir care conține un nume calificat prin zonă de nume, reprezentând numele extins al primului nod din argumentul `node-set`, în ordinea documentului. Un nume calificat prin zonă de nume este numele nodului, plus prefixul zonei de nume asociat URI-ului extins al zonei de nume. Dacă argumentul lipsește, atunci `node-set` este nodul curent sau context.

Funcţii referitoare la şiruri

Funcţiile referitoare la şiruri permit manipularea de şiruri din conţinutul nodurilor şi atributelor:

- ***string*** **string(*object?*)** – Funcţia `string()` converteşte un obiect într-un şir. Dacă un nod obiect conţine un număr, atunci el este convertit la şirul echivalent, dar această funcţie nu este menită să convertească numere care să fie apoi prezentate unui utilizator. Dacă obiectul este o listă de noduri, atunci este returnată valoarea şir a primului nod din listă, în ordinea documentului. Dacă obiectul parametru lipseşte, atunci este returnată valoarea şir a nodului curent sau context. Dacă nodul este vid, atunci este returnat şirul vid.
- ***string*** **concat(*string*, *string*, *string**)** – Funcţia `concat()` returnează concatenarea argumentelor sale. Aceasta este una dintre cele mai simple operaţii cu şiruri, permiţând alegerea din oricâte locuri a unor părţi individuale de şir şi asamblarea uneia sau a mai multor părţi la un moment dat.
- ***boolean*** **starts-with(*string*, *string*)** – Funcţia `starts-with()` returnează true (adevărat) dacă primul şir argument începe cu cel de-al doilea şir argument, iar în caz contrar returnează `false` (fals).
- ***boolean*** **contains(*string*, *string*)** – Funcţia `contains()` returnează adevărat dacă primul şir argument îl conţine pe cel de-al doilea şir argument, iar în caz contrar returnează fals.
- ***string*** **substring-before(*string*, *string*)** – Funcţia `substring-before()` returnează acel subşir al primului şir argument care precedă prima apariţie a celui de-al doilea şir argument în primul şir argument, sau returnează şirul vid dacă primul şir argument nu îl conţine pe cel de-al doilea şir argument. Aceasta vă permite sa editaţi şirul sursă pentru a elimina din el un capăt nedorit, înainte de a-l folosi.
- ***string*** **substring-after(*string*, *string*)** – Funcţia `substring-after()` returnează acel subşir al primului şir argument care urmează după prima apariţie a celui de-al doilea şir argument în primul şir argument, sau returnează şirul vid dacă primul şir argument nu îl conţine pe cel de-al doilea şir argument. Aceasta vă permite sa editaţi şirul sursă pentru a elimina din el un început nedorit, înainte de a-l folosi.
- ***string*** **substring(*string*, *number*, *number?*)** – Funcţia `substring()` returnează acel subşir al primului şir argument care începe la poziţia specificată de cel de-al doilea argument şi are lungimea specificată de cel de-al treilea argument. Aceasta vă permite să editaţi şirul sursă pentru a selecta dintr-un şir un subşir de lungime cunoscută, înainte de a-l folosi.
- ***number*** **string-length(*string?*)** – Funcţia `string-length()` returnează numărul de caractere din şir. Dacă argumentul este omis, el se referă la nodul context convertit la un şir, adică valoarea şir a nodului context.
- ***string*** **normalize-space(*string?*)** – Funcţia `normalize-space()` returnează şirul argument cu spaţiile albe normalizate prin eliminarea spaţiilor albe anterioare şi posterioare şi înlocuirea secvenţelor de caractere spaţii albe cu un singur spaţiu. Caracterele spaţii albe sunt identice cu cele permise de Recomandarea XML 1.0.
- ***string*** **translate(*string*, *string*, *string*)** – Funcţia `translate()` returnează primul şir argument cu apariţiile de caractere în cel de-al doilea şir argument înlocuite cu caracterele de pe poziţia corespunzătoare din cel de-al treilea şir argument. Aceasta vă permite să editaţi şirul sursă pentru a schimba majusculele în litere mici sau pentru orice altă transformare asemănătoare.

Funcţii booleene

Funcţiile booleene realizează diverse teste sau operaţii booleene asupra argumentelor lor şi returnează o valoare bazată pe test sau pe operaţie:

- ***boolean*** **boolean(*object*)** – Funcţia `boolean()` converteşte argumentul său la o valoare booleeană. Dacă obiectul este zero, vid sau de lungime zero, atunci valoarea booleeană este fals, altfel este adevărat.
- ***boolean*** **not(*boolean*)** – Funcţia `not()` returnează adevărat dacă argumentul ei este fals, şi fals în caz contrar. Cu alte cuvinte, această funcţie returnează negarea valorii sale booleene.
- ***boolean*** **true()** – Funcţia `true()` returnează adevărat.
- ***boolean*** **false()** – Funcţia `false()` returnează fals.
- ***boolean*** **lang(*string*)** – Funcţia `lang()` returnează adevărat sau fals, după cum limba nodului context, aşa cum este aceasta specificată de atributele xml:lang, este aceeaşi cu limba specificată de către şirul argument sau este o sublimbă a acesteia.

Funcţii referitoare la numere

Funcţiile referitoare la numere efectuează diverse operaţii numerice asupra argumentelor lor, realizând conversii dacă este nevoie:

- ***number*** **number(*object?*)** – Funcţia `number()` converteşte argumentul ei la un număr.
- ***number*** **sum(*node-set*)** – Funcţia `sum()` returnează suma, pentru fiecare nod din argumentul `node-set`, rezultatelor obţinute prin conversia în numere a valorilor şir ale nodurilor.
- ***number*** **floor(*number*)** – Funcţia `floor()` returnează cel mai mare număr care nu este mai mare decât argumentul şi este un întreg.
- ***number*** **ceiling(*number*)** – Funcţia `ceiling()` returnează cel mai mic număr care nu este mai mic decât argumentul şi este un întreg.
- ***number*** **round(*number*)** – Funcţia `round()` returnează numărul cel mai apropiat de argument şi care este un întreg.

Utilizarea predicatelor XPath

Asemenea oricărui API de programare, chiar dacă domeniul de aplicabilitate este restrâns în mod intenţionat, funcţiile XPath pot fi combinate şi utilizate în aproape o infinitate de moduri. În plus, multe dintre funcţiile mai complexe sunt mult mai utile în combinaţie cu alte API-uri, sau cu XSLT, sau cu o altă aplicaţie care manipulează efectiv documentul. Cu toate că, de exemplu, este foarte comod să poţi să adresezi subşiruri dintr-un nod text, dacă nu veţi folosi acea informaţie pentru a modifica documentul sau un alt document care utilizează acel subşir, totul este doar un exerciţiu teoretic.

Deci o imaginaţie activă este o parte importantă din decizia asupra modului de utilizare a aptitudinilor predicatelor. Fără a vă hotărî mai întâi ce doriţi să faceţi, cunoaşterea a ceea ce puteţi face este oarecum inutilă. Dar, în acelaşi timp, cunoaşterea a ceea ce puteţi face este un impuls dat imaginaţiei şi poate fi de mare ajutor atunci când decideţi dacă un lucru este sau nu posibil.

Priviți următorul exemplu simplu pentru a lămuri principiile generale. Funcțiile referitoare la șiruri pot fi utilizate pentru a realiza manipulări simple ale nodurilor text într-un document XML. Să presupunem că aveți un fragment de document care arată astfel:

```
<data-cumparare id="pd123">04/17/2000</data-cumparare>
```

Pentru a extrage anul, puteți folosi următoarea combinație de comenzi:

```
substring-after(substring-after(string(id("pd123")),"/"),"/")
```

Aceasta va returna șirul „2000". Ceea ce doriți să faceți cu informația este treaba dumneavoastră. Funcția `id()` este utilizată pentru a extrage nodul având valoarea „pd123" și apoi se utilizează funcția `string()` pentru a converti la un șir conținutul acelui nod. Apoi, funcția `string-after()` a fost utilizată de două ori pentru a „tăia" prima dată „04/" și apoi „17/" din șirul rezultat, dând ca rezultat anul „2000" în format șir. Evident, în acest moment, puteți împacheta totul în

```
string(number("2000") div 2)
```

dacă ați preferat mileniul trecut celui actual.

Probleme cu ancorele și cu legăturile HTML

Legăturile tradiționale și pointerii sunt, pe de-o parte, mult prea simple pentru a se adapta modalităților în care își pot imagina oamenii utilizarea acestora. Pe de altă parte, ele sunt mult prea complexe. În HTML și în XHTML există câteva metode de a lega un obiect sau o locație, utilizând un număr mare de nume de atribute pentru a descrie hiperlegătura efectivă. Aceste metode sunt următoarele:

- Ca hiperlegătură într-o ancoră, sau într-o etichetă `<a>`, care poate fi utilizată atât ca element de legătură cât și ca țintă, o combinație confuză de funcții, cu un atribut *href* atunci când este utilizată ca o legătură și un nume sau o referință id atunci când este utilizată ca țintă a unei legături.
- Ca hiperlegătură într-o imagine, sau într-o etichetă `<img>`, care are încorporate de fapt trei legături posibile: atributele *src*, *usemap* și *longdesc*. Acest tip de legătură indică o formă primitivă de transcludere care a fost inclusă de caracteristicile XML mult mai generale, descrise în XLink.
- Ca hiperlegătură într-o legătură, sau într-o etichetă `<link>`, cu un atribut *href*. Aceste legături pot indica relații între documente HTML și pot fi reprezentate sau nu într-un anumit mod de un agent utilizator. Maniera în care sunt utilizate aceste legături nu este foarte uniformă și este foarte dependentă de browser.
- Ca un fragment de hiperlegătură într-o cale, sau într-o etichetă `<base>`, cu un atribut *href*. Fragmentul identificat de atributul *href* este utilizat pentru a construi un URL complet.
- Ca hiperlegătură într-o suprafață, sau într-o etichetă `<area>`, cu un atribut *href*. Aceasta este o formă foarte primitivă de legare indirectă sau externă, care a fost inclusă în caracteristicile XML mult mai generale, descrise în XLink.
- Ca hiperlegătură într-un cadru, sau într-o etichetă `<frame>`, cu atributele *src* și *longdesc*. Acest tip de legătură indică o formă primitivă de transcludere, care a fost inclusă în caracteristicile XML mult mai generale, descrise în XLink.
- Ca hiperlegătură într-un cadru in-line, sau într-o etichetă `<iframe>`, cu atributele *src* și *longdesc*. Acest tip de legătură indică o formă primitivă de transcludere, care a fost inclusă în caracteristicile XML mult mai generale, descrise în XLink.

- Ca hiperlegătură într-un applet (miniaplicație), sau într-o etichetă `<applet>`, cu atributele *codebase* și *archive*. Ambele reprezintă sursa resurselor utilizate într-o manieră asemănătoare cu ceea ce poate fi făcut acum mult mai uniform, prin transcluderea în stil XML, care a fost inclusă în caracteristicile XML mult mai generale, descrise în XLink.
- Ca hiperlegătură într-un obiect, sau într-o etichetă `<object>`, cu atributele *classid*, *codebase*, *data*, *archive* și *usemap*. Toate acestea sunt forme primitive de transcludere, care au fost incluse în caracteristicile XML mult mai generale, descrise în XLink.
- Ca hiperlegătură într-un parametru, sau într-o etichetă `<param>`, cu atributul *ref*. Din nou, aceasta este o formă primitivă de transcludere.
- Ca hiperlegătură într-un citat sau bloc citat, sau în etichete `<q>` și `<blockquote>`, fiecare având atributul *cite*. Această legătură poate să fie sau să nu fie implementată de către agentul utilizator HTML.
- Ca hiperlegătură într-o zonă de inserare sau ștergere, sau în etichete `<ins>` și `<del>`, fiecare având atributul *cite*. Această legătură poate să fie sau să nu fie implementată de către agentul utilizator HTML.
- Ca hiperlegătură într-un antet, sau într-o etichetă `<head>`, cu un atribut *profile*. Această legătură poate să fie sau să nu fie implementată de către agentul utilizator HTML.
- Ca hiperlegătură într-un corp, sau într-o etichetă `<body>`, cu un atribut *background* care implementează o încorporare în stil transcludere a unei imagini într-un document.
- Ca hiperlegătură într-un script, sau într-o etichetă `<script>`, cu un atribut *input*.
- Ca hiperlegătură într-o intrare, sau într-o etichetă `<input>`, cu un atribut *usemap*.
- Ca hiperlegătură într-un formular, sau într-o etichetă `<form>`, cu un atribut *action*.

Sintaxa utilizată pentru legarea la sursa externă nu prea are uniformitate, în afară de utilizarea uneori a atributelor *href* pentru a implica un fel de legătură hipertext, atributului *src* rămânându-i să implice un anumit tip de transcludere. Dar tabloul uimitor de nume de atribute care se referă la locațiile legate este predispus la erori și excesiv de complex.

De exemplu, o legătură externă într-un element ancoră HTML (`<a>`) arată astfel

```
<a href="URI"></a>
```

dar o legătură similară într-o hartă-imagine este referită indirect prin

```
<img src="URI-1" usemap="URI-2" ismap>
```

Diferența dintre numele atributelor utilizate pentru transcludere, în maniera unei etichete HTML `img`, și cele utilizate pentru hiperlegăturile utilizabile poate avea sens într-o lume de dinainte de XLink. Dar acum nu prea mai sunt motive pentru aceste încercări artificiale de a predefini un comportament, deoarece XLink generalizează complet sintaxa legării și separă tipurile diferite de legături prin atribute specifice, care descriu tipul efectiv al legăturii care se stabilește. În locul unei casete de instrumente cu legături specializate având scopuri specifice, noul kit cu instrumente pentru legare permite ca o mână de instrumente să realizeze operații multiple.

Multe dintre numele speciale de atribute au fost încercări de a identifica natura exactă a legăturii sau de a-i spune procesorului ce ar trebui (sau ar putea) să realizeze cu aceasta. Multe dintre numele *speciale* de legături au contribuit, de asemenea, la redundanță, deci

recuperarea în urma erorilor era mai ușoară. Multe din atributele de legare nu au fost implementate niciodată într-un browser comercial, de exemplu atributul *cite* pentru citate și marcajele de modificare a textului, sau atributul *longdesc*.

O metodă uniformă de descriere a legăturilor și o metodă prin care pot fi prezentate opțiunile vor reprezenta un mare pas înainte, permițând autorilor de browsere să implementeze efectiv întreaga recomandare, și nu doar părți care le-au plăcut sau au văzut că sunt ușor de respectat.

Studii de caz

Deoarece nu puteți face deocamdată prea multe cu o relație sau cu un predicat XPath, excepție făcând indicarea unui loc, în afară de acest fapt simplu nu puteți spune prea multe despre ele. Diversele funcții definite de XPath sunt utilizate numai ca metode de obținere a indicilor sau de determinare a condițiilor care vă permit să indicați mai precis o locație.

XLink și XPointer

CAPITOLUL 9

În acest capitol

Noul kit cu instrumente de legare XML
Utilizarea XLink, limbajul de legare extensibil (Extensible Linking Language)
Descifrarea sintaxei XLink
XPointer
Depanarea XLink și XPointer

Noul kit cu instrumente de legare XML

Acest capitol examinează cele două mecanisme de legare utilizate de XML, XLink și XPointer. Primul este destinat creării efective și traversării legăturilor, în timp ce al doilea este utilizat pentru a adresa fragmente de documente XML utilizate pentru a construi documente noi din părți ale unor documente vechi, prin extinderea posibilităților de adresare ale XPath. XPointer adaugă compararea șirurilor și poziționarea absolută în funcție de cursor sau de un domeniu în elemente, deci este posibilă o adresare precisă, chirurgicală, pentru a extrage un singur cuvânt la fel de ușor ca un interval care conține o porțiune mare a unui document extern.

Atenție Specificația XLink este încă în stadiul de schiță de lucru, dar se îndreaptă cu repeziciune spre statutul de Recomandare. Vom trata schița actuală ca și când ar fi versiunea finală, dar vă rog să rețineți că unele detalii se pot modifica. Vă rog să consultați versiunea actuală de XLink la `http://www.w3.org/tr/`.

XLink extinde cunoscuta etichetă ancoră de legare HTML (și alte etichete cu funcționalitate de legare) cu numeroase adăugiri la acea tehnologie de bază care, trebuie subliniat, este temelia și inima World Wide Web-ului hiperlegat. La repertoriul ancorelor HTML de bază, legături, ancore și URL-uri, XLink adaugă:

- Legături la destinații multiple
- Legături bidirecționale
- Legături care indică la documente numai pentru citire (utilizând XPath)
- Legături care sunt etichetate cu titluri descriptive, clarificând astfel modul de utilizare a legăturii
- Legături care se extind pe loc (un proces numit *transcludere*) pentru a îngloba un conținut extern într-un document XML, așa cum sunt acum înglobate imaginile

- Legături care sunt parcurse automat atunci când este încărcat documentul. Aceasta poate permite includerea de conținut extern sau înlocuirea unei pagini cu alta.

În timp ce HTML acceptă transcluderi limitate, sub forma etichetelor `<img>` și `<object>`, XLink permite ca orice entitate externă să fie utilizată în același mod.

Notă Specificația actuală a W3C referitoare la trioul XPath/XPointer/XLink este de fapt mai puțin puternică în unele privințe decât specificația XSL actuală. Cu toate că aceasta se va modifica în mod sigur, fiți conștient că nu există un singur mod de a lega documente în alaiul crescând de standarde XML.

XPointer este extremitatea activă a lui XLink, bazată pe XPath, dar adăugând câteva îmbunătățiri proprii. XPointer poate adresa atât intervale dintr-un document cât și locații izolate, chiar și șiruri din document și chiar fără nici o etichetă specială care să fie trecută în document de către autor. Acolo unde XPath se bazează pe *noduri*, locații speciale din document identificate prin etichete, XPointer permite mutarea locației exacte a pointerului față de centru, pentru adresarea caracterelor individuale, a șirurilor și chiar a intervalelor care trec de limitele elementului.

Împreună cu XPath, XPointer asigură că pointerii sunt lizibili și utilizabili de ființele umane, spre deosebire de contoarele de octeți și de indicii hash, înțelese de calculator, dar pe care un om nu le-ar putea înțelege fără ajutor. XPointer are chiar o sintaxă specială, numită *adresare numai prin nume*, proiectată să permită indicarea în documente HTML utilizând sintaxa simplă `URI#nume-ancora` în loc de `URI#xpointer(id("nume-ancora"))`, pentru a accepta cantitatea mare de texte moștenite, disponibile pe Web.

Utilizarea XLink, limbajul de legare extensibil (Extensible Linking Language)

XLink sosește. Deoarece XML nu are un set de etichete fixat, este necesară o metodă de declarare de legături asociate oricărui element. Deoarece nu vă puteți baza pe informația contextuală care să vă spună ce să faceți cu legătura, toate diferențele subtile de comportament văzute în exemplele HTML de tipuri de legături de mai sus trebuie să fie înlocuite cu descrieri explicite. Și deoarece toată lumea a fost de acord că este nevoie de caracteristici suplimentare, o varietate de tipuri de legături, pe lângă cele utilizate de HTML, au fost adăugate la acest amestec. Exemplele includ legături care pot fi parcurse în oricare direcție și legături care oferă mai multe destinații, care pot fi alese de utilizator la selectarea legăturii. Deci o singură legătură se poate comporta ca o fereastră autoderulantă sau pop-up.

Și, în ciuda faptului că poate părea complex, în unele privințe este de fapt o simplificare a statu-quo-ului actual. În locul unei liste lungi de elemente specifice și atribute asociate cu comportament de legătură sau asemănător cu acesta, acum există un mecanism care poate fi utilizat fie pentru a dubla practicile existente, fie pentru a extinde posibilitățile comportamentului de legare în moduri dificil de închipuit (în afară de cazul în care veniți din lumea HyTime, modelul original și impulsul care stă la baza standardelor de legare XML).

Pe de altă parte, XLink nici măcar nu încearcă să acopere relațiile complexe dintre obiectele legate, așa cum fac atributele HTML `archive` sau `usemap`. Mai este nevoie de multă muncă.

Secțiunile următoare studiază caracteristicile specifice XLink care diferă substanțial de tehnologiile de legare HTML anterioare.

Accesarea întregului document

Utilizând XPath și XPointer, XLink este capabil să acceseze orice parte a documentului, nu numai rădăcina și părțile numite diferit sau ancorate. Aceasta înseamnă că un citat, de exemplu, poate fi inserat într-un document prin *transcludere*, copiind din punct de vedere logic conținutul unui fișier extern într-un document gazdă, și în același timp poate fi și atribuit. Deci dacă doriți să găsiți sursa citatului, urmăriți pur și simplu o legătură, poate una dintre cele care au fiecare legătură etichetată descriptiv, până la corpul efectiv al textului de unde a fost extras citatul. Sau, alternativ, un cercetător sau un student, chiar fără a avea acces de scriere la textul care utilizează citatul, ar putea insera din exterior o astfel de legătură, echivalentul tehnologic al etichetelor adezive, atunci când un document extern este vizualizat dintr-un document gazdă. Deoarece legăturile multiple sunt permise, oricâte note se pot referi la aceeași locație sau același articol, iar diferiți cercetători pot utiliza propriile lor documente gazdă locale pentru a adnota același document sursă cu propriile lor cercetări și concluzii.

Atenție XLink, asemenea cadrelor, permite transcluderea necontrolată a conținutului cu drept de autor de pe alte situri sau aparținând altor autori, dacă este utilizat necorespunzător. Utilizarea fără permisiune a imaginilor sau a cuvintelor cu drept de autor ale altor oameni este considerată furt, chiar dacă indicați sursa pentru a evita stigmatul plagiatului. Au fost mai multe procese câștigate împotriva siturilor care și-au însușit conținuturi în mod ilegal, deci asigurați-vă că obțineți în prealabil permisiunea, sau cereți sfatul unui avocat specializat în drepturi de autor înainte de a încorpora conținut extern în propriul dumneavoastră sit.

În ciuda faptului că *transcluderea* poate fi un cuvânt nou pentru dumneavoastră, conceptul este foarte simplu. Aceasta înseamnă că un fișier extern este inclus într-un document ca și când ar exista efectiv în el, cu toate că în realitate el rămâne separat.

XLink face posibile astfel de lucruri pentru aproape orice conținut, nu numai pentru imagini. Dacă un fișier text este transclus într-un document gazdă, puteți căuta textul în documentul gazdă ca și când ar fi fost prezent în acesta tot timpul. În ciuda faptului că acest document virtual mai cuprinzător este alcătuit rapid, există numai pentru moment și va dispărea atunci când veți termina vizualizarea și interacțiunea cu el, se comportă ca și când ar fi real.

Notă Dată fiind existența pe Internet a unui corp mare de text cum ar fi proiectul Gutenberg – precum și colecții de documente mai specializate și replici grafice ale unor incunabule și manuscrise medievale – oportunitățile de învățare on-line se înmulțesc fără limită. Regulile existente pentru studiile electronice pot fi găsite în pagina de bază Text Encoding Initiative Consortium (Consorțiul Inițiativei de Codificare a Textului), la `http://www.tei-c.org/`.

Unul dintre cele mai interesante avantaje ale XLink este capacitatea sa de a crea legături în fișiere multimedia non-text. Deci puteți face legături nu numai la text, legături cu o granulozitate mult mai bună decât pot furniza legăturile HTML existente.

Desigur, XLink nu se rezumă numai la învățare, iar posibilitățile pentru tipuri noi de documente interactive, fără complexitatea implementărilor brevetate din Dynamic HTML, extind potențialul Web-ului și oferă un domeniu mai larg pentru proiectanții de documente XML de a explora frontierele creativității și puterii.

Tipuri de legare XLink

XLink face întrucâtva artificială și redundantă diferența dintre legăturile simple și legăturile extinse, legăturile simple fiind echivalente cu ancorele simple stil HTML (*locator*) și legăturile cu imagini (). Legăturile extinse oferă proiectanților întreaga putere a mecanismului XLink, în ciuda faptului că ele nu sunt inteligibile imediat pentru browserele HTML moștenite – ca de altfel nici noile legături simple.

În afară de compatibilitatea îndoielnică cu versiunile anterioare și de capacitatea de a utiliza familiarul atribut `href=URI` la ceva care seamănă cu o ancoră pentru a construi o legătură, nu par să existe motive pentru a utiliza sintaxa simplă. Iar dificultatea migrării unei legături de la vechea sintaxă simplă la noua versiune extinsă pare să dea argumente pentru a începe de la zero cu noua sintaxă, astfel încât legăturile să evolueze cu ușurință o dată cu situl. Secțiunile următoare furnizează o privire generală rapidă asupra celor două tipuri principale de legături, simple și extinse.

Legături simple

O *legătură simplă* presupune o resursă locală și una la distanță, iar legătura poate fi traversată numai într-o singură direcție și este întotdeauna in-line. *In-line* înseamnă numai că există o resursă locală care participă la legătură.

Atenție În exemplul următor, DTD-ul poate să fie sau să nu fie citit pentru a determina valorile prestabilite ale atributelor, deci în majoritatea documentelor ar trebui să setați explicit fiecare atribut. Acest exemplu sare peste această prevedere pentru a simplifica codul XML implicat.

O legătură simplă poate fi construită utilizând variante ale următoarei definiții a unui DTD și ale instrucțiunilor XML:

```
<!ELEMENT simpleLink ANY>
<!ATTLIST simpleLink
  xlink:type     (simple)              #FIXED "simple"
  xlink:href     CDATA                 #REQUIRED
  xlink:role     NMTOKEN               #FIXED "anyPredefinedRole"
  xlink:title    CDATA                 #IMPLIED
  xlink:show     (new
                 |replace
                 |embed
                 |undefined)           #FIXED "replace"
  xlink:actuate (onLoad
                 |onRequest
                 |undefined)           #FIXED "onRequest" >
```

Această definiție particulară funcționează ca echivalent al unui element ancoră HTML. Pentru un comportament care să imite un element imagine HTML, redefiniți atributul `xlink:actuate` ca `onLoad`, iar atributul `xlink:show` ca `embed`. Și poate veți dori să declarați și modelul de conținut ca fiind `EMPTY`.

Pentru un comportament tipic de legătură hipertext puteți folosi o legătură simplă ca aceasta:

```
<simplelink href="URI">Ceva continut</simplelink>
```

Sau, dacă ați definit legătura simplă ca un element vid cu sursa `show="embed"` și cu comportament `actuate="onLoad"`, adică transclude automat conținutul referit atunci când este încărcat documentul gazdă, atunci o puteți utiliza astfel:

```
<simplelink href="URI" />
```

Evident, atributele corespunzătoare, `show` și `actuate`, vor fi definite în `ATTLIST`-ul `simplelink`, așa cum se arată în secțiunea „Structura elementelor XLink".

Legături extinse

Legăturile extinse pot conține orice număr de resurse, nefiind necesar ca vreuna dintre ele să fie locală. Dacă nu există resurse locale, legătura este și ea externă, cu alte cuvinte utilizează numai documente și locații externe și este localizată acolo și nu în documentul gazdă. În caz contrar, legătura este in-line.

Atenție Ca și în exemplul anterior, al unei legături simple, DTD-ul pentru exemplul următor poate fi citit sau nu pentru a determina valorile prestabilite ale atributelor. În interesul simplității, în acest exemplu atributele nu sunt setate explicit.

Construirea unei legături extinse este mai complexă. Totuși, posibilitatea de a include mai mult de o resursă într-o singură legătură oferă mai mult decât funcționalitatea unei legături obișnuite, iar extinderea după dorință a legăturii o face cu mult mai versatilă. În ciuda faptului că o legătură simplă nu necesită un anumit conținut, o legătură extinsă conține mai multe elemente conținut care contribuie la funcționalitatea totală a legăturii. DTD-ul următor poate fi utilizat pentru a defini profilul de conținut al unei legături extinse echivalentă cu legătura simplă din secțiunea precedentă:

```
<!ELEMENT extendedlink (locatorElement, arcElement, resourceElement)>
<!ATTLIST extendedlink
  xlink:type     (extended)      #FIXED "extended" >

<!ELEMENT locatorElement EMPTY>
<!ATTLIST locatorElement
  xlink:type     (locator)       #FIXED "locator"
  xlink:href     CDATA           #REQUIRED
  xlink:role     (remote)        #FIXED "remoteRoleType"
  xlink:title    CDATA           #IMPLIED >

<!ELEMENT arcElement EMPTY>
<!ATTLIST arcElement
  xlink:type     (arc)           #FIXED "arc"
  xlink:from     (local)         #FIXED "local"
  xlink:to       (remote)        #FIXED "remote"
  xlink:show     (new
                 |replace
                 |embed
```

```
                |undefined)     #IMPLIED
  xlink:actuate (onLoad
                |onRequest
                |undefined)     #IMPLIED >

<!ELEMENT resourceElement ANY>
<!ATTLIST resourceElement
  xlink:role    (local)         #FIXED "local" >
```

În acest exemplu, elementul arc utilizează atributele predefinite `xlink:to` și `xlink:from`. Legătura extinsă o puteți utiliza astfel:

```
<extendedlink>
  <locatorElement xlink:href="URI" />
  <arcElement />
  <resourceElement>click here</resourceElement>
</extendedlink>
```

Sunt posibile și tipuri de legături mult mai complexe, dar de fapt, codul nu este cu mult mai complicat decât pentru legătura simplă de mai sus. Posibilitatea extinderii ulterioare a legăturii face această sintaxă mult mai versatilă. Modelul de conținut al legăturii extinse poate fi făcut mai general, iar punctele fixe „la" și „de la" ale arcului pot fi definite, deci în aceeași legătură pot fi incluse mai multe resurse:

```
<!ELEMENT extendedlink (locatorElement | arcElement | resourceElement)*>
<!ATTLIST extendedlink
  xlink:type    (extended)      #FIXED "extended" >

<!ELEMENT locatorElement EMPTY>
<!ATTLIST locatorElement
  xlink:type    (locator)       #FIXED "locator"
  xlink:href    CDATA           #REQUIRED
  xlink:role    NMTOKEN         #IMPLIED
  xlink:title   CDATA           #IMPLIED>

<!ELEMENT arcElement EMPTY>
<!ATTLIST arcElement
  xlink:from    NMTOKEN         #IMPLIED
  xlink:to      NMTOKEN         #IMPLIED
  xlink:type    (arc)           #FIXED "arc"
  xlink:show    (new
                |replace
                |embed
                |undefined)     #IMPLIED
  xlink:actuate (onLoad
                |onRequest
                |undefined)     #IMPLIED >

<!ELEMENT resourceElement ANY>
<!ATTLIST resourceElement
  xlink:role    (local)         #FIXED "local" >
```


În acest exemplu, elementul arc utilizează un tip de arc predefinit, `xlink:type`, pentru a identifica ce fel de element XLink este, dar lasă nedefinite valorile pentru `xlink:to` și `xlink:from` astfel încât ele pot avea valori asociate lor chiar în documentul XML. Legătura extinsă o puteți utiliza astfel:

```
<extendedlink>
  <locatorElement xlink:role="parent" xlink:href="URI-1" xlink:title="Doc-1" />
  <locatorElement xlink:role="child" xlink:href="URI-2"  xlink:title="Doc-2" />
  <locatorElement xlink:role="child" xlink:href="URI-3"  xlink:title="Doc-3" />
  <arcElement xlink:from="parent" xlink:to="child" />
  <arcElement xlink:from="child" xlink:to="parent" />
  <resourceElement>Link to related file here.</resourceElement>
</extendedlink>
```

Nu există un comportament predefinit asociat acestui tip de legătură care prezintă o opțiune posibilă de legătură între resursa părinte și oricare dintre cei doi copii, sau dintre oricare dintre cei doi copii și o resursă părinte. Un agent utilizator poate oferi un meniu pop-up de opțiuni sau orice altă prezentare cu rezultate finale asemănătoare.

Descifrarea sintaxei XLink

Un XLink este definit de obicei într-un DTD XML printr-o serie de atribute, majoritatea lor fiind opționale. Valoarea primului atribut din lista următoare, type, este transportată de asemenea de un set corespunzător de elemente predefinite având același nume cu valorile atributului type în zona de nume XLink. Aceasta înseamnă că puteți utiliza XLink-uri în documentele proprii fără a defini elementele de legătură în DTD-ul dumneavoastră prin utilizarea etichetelor XLink predefinite, o simplificare comodă.

Un element XLink poate fi orice element XML având inclus un atribut XLink type în ATTLIST-ul său, ori un element XLink propriu-zis, după cum este definit de DTD-ul XLink în zona de nume XLink. Atunci când, în acest capitol, ne referim la un element XLink înțelegem ambele posibilități, în mod interschimbabil, dacă nu ne limităm explicit la una dintre ele. În afară de type, toate atributele de mai jos pot fi asociate fie cu elemente XLink propriu-zise, fie cu elemente din propriul dumneavoastră DTD care au fost definite ca elemente XLink prin adăugarea unui atribut XLink type. Atributele XLink sunt prezentate în lista următoare:

- `type` – Are una din următoarele șapte valori:
  - `simple` asemănător cu un element ancoră HTML, perechea de etichete `<a href="URI"></a>`. Atributul `simple` corespunde etichetei cu același nume din zona de nume XLink. În ciuda faptului că o legătură `extended` este capabilă să aibă exact același comportament, tipul special `simple` permite o sintaxă care seamănă puțin cu o legătură HTML, pentru ușurința utilizării.
  - `extended` vă permite să includeți mai multe elemente de legătură într-un element legătură extinsă. Atributul `extended` corespunde etichetei cu același nume din zona de nume XLink.
  - `locator` elemente care conțin un atribut `locator` sunt menite să facă parte dintr-un element legătură extinsă. Ele sunt echivalente cu conținutul unei hiperlegături în HTML, adică vă permit să vedeți unde este localizată legătura în document și să o

selectați. Valoarea atributului locator corespunde etichetei cu același nume din zona de nume XLink.

- arc definește comportamentul de traversare în legăturile extinse. Valoarea atributului arc corespunde etichetei cu același nume din zona de nume XLink.
- resource definește o parte vizibilă de legătură care participă în orice legătură XLink. Valoarea atributului resource corespunde etichetei cu același nume din zona de nume XLink.
- title definește un șir lizibil asociat legăturii și care poate fi afișat de un agent utilizator. Valoarea atributului title corespunde etichetei cu același nume din zona de nume XLink.
- none indică faptul că acel element nu are nici o semnificație XLink. Valoarea atributului none corespunde etichetei cu același nume din zona de nume XLink.

În cadrul zonei de nume XLink, aceste șapte valori sunt definite și ca elemente și pot fi utilizate direct, de exemplu:

```
<xlink:simple href="URI" attributes />
```

Comportamentul acestor șapte tipuri de legături este cercetat în detaliu mai departe în acest capitol. Un atribut type este necesar în toate tipurile de elemente XLink.

- href – Atributul care definește unde este localizată resursa țintă. Atributul href nu este permis pentru elementele de tipul extended, arc, resource și title și este obligatoriu pentru elementele locator.
- to – Un capăt al unui arc sau al unei legături care conectează două resurse. Valoarea ar trebui să fie un IDREF care să refere resursa role. Atributul to nu este permis pentru elementele de tipul simple, extended, locator, resource și title.
- from – Celălalt capăt al unui arc sau al unei legături care conectează două resurse. Valoarea ar trebui să fie un IDREF care să refere resursa role. Atributul from nu este permis pentru elementele de tipul simple, extended, locator, resource și title.

Notă Sintaxa lui to și a lui from mi se pare puțin neclară, deoarece specificația permite un singur from sau to pentru a descrie legătura dintre mai multe instanțe de documente având același nume role. Deci, în ciuda faptului că role nu este un id, în sensul tipic al XML care implică unicitatea, referința la aceste instanțe role multiple implică faptul că ele sunt unice.

- show – În legături simple, ia trei valori:
  - new creează o fereastră nouă în care să afișeze conținutul. Aceasta ar trebui utilizată cu precauție sau deloc.
  - replace face ceea ce face acum în mod tipic o hiperlegătură, adică înlocuiește conținutul ferestrei browserului cu conținut nou.
  - parse este destinat să preia conținutul resursei de la distanță și să îl insereze într-un document, ca și când ar fi fost inclus acolo de la început.

Atributul show nu este permis pentru elementele de tipul locator, resource și title.

- actuate – În legături simple, poate lua două valori:
  - user – permite utilizatorului să aleagă când să activeze o legătură.
  - auto – activează automat legătura.

Atributul actuate nu este permis pentru elementele de tipul locator, resource și title.

Atenție Legăturile care sunt urmate sau actualizate automat – așa cum poate fi făcut acum în HTML prin tehnicile client pull sau server push (pentru a forța actualizarea unei pagini de către agentul utilizator, respectiv de către server) – sunt adesea dăunătoare. Înlocuirea automată a unor pagini întregi este ostilă față de persoanele care se pot încurca din cauza acestei modificări sau nu pot utiliza funcția back (înapoi la pagina anterioară) destul de repede pentru a scăpa din strânsoarea acestor scheme greșit concepute. Cititoarele de ecran încep deseori citirea din partea de sus a paginii de fiecare dată când pagina este reîmprospătată, în ciuda faptului că această „pâlpâire" acustică poate fi invizibilă pentru un utilizator fără probleme de vedere. În același mod, invocarea automată a paginilor new este un obicei înfiorător atunci când este utilizată de către proiectanții Web pentru a face siturile lor „lipicioase" în dauna utilizatorilor cu vedere, memorie sau dexteritate limitate. Utilizarea legăturilor automate pentru înlocuirea completă a conținutului sau crearea unei ferestre noi este contrară naturii WAI; singura utilizare justificată ar fi analizarea și inserarea de conținut transclus, cum ar fi echivalentul XLink al etichetelor HTML img.

- role – Conține un șir menit să descrie scopul legăturii, pentru prelucrarea într-un anumit fel a acesteia. Cu toate că șirul poate fi lizibil pentru om, el nu este destinat afișării. Un element arc poate utiliza atributele role asociate unei resurse pentru a identifica punctele de capăt ale arcului și direcția urmată pentru parcurgerea acestuia. Atributul role nu este permis pentru elementele de tipul title.
- title – Conține un șir menit să afișeze o descriere lizibilă pentru om a scopului legăturii. În ciuda faptului că WAI sugerează ca legătura în sine să conțină text lizibil pentru om, care să descrie legătura, și nu expresii non-informative cum ar fi „click here" (clic aici), faptul că orice element poate fi utilizat ca un XLink înseamnă că ar trebui să fie disponibile mijloace alternative de transmitere a acestei informații. Atributul title este un mecanism uniform pentru furnizarea unui text explicativ în orice situație.

Atributele precedente sunt utilizate în elemente, ca orice atribut XML, iar atributul xlink:type este cheia funcționalității elementului. Pentru ușurința utilizării, XLink predefinește, de asemenea, elemente cu nume identice cu fiecare tip, deci nu trebuie să definiți mecanisme XLink în DTD-ul dumneavoastră, dar puteți utiliza elemente XLink direct din zona de nume XLink. Subsecțiunea următoare dezbate definirea într-un DTD a elementelor de structură pentru elementele XLink.

Structura elementelor XLink

Această secțiune prezintă explicit atributele pe care le poate avea fiecare tip de element XLink. Nu este necesar fiecare atribut, iar atributul type poate fi înlocuit cu elemente explicite, predefinite în zona de nume XLink. În această situație, le puteți utiliza în documentele dumneavoastră în mod direct și nu trebuie definite XLink-uri în DTD-ul dumneavoastră. Dar pentru internaționalizare și localizare, posibilitatea de a defini XLink-uri cu nume sugestive poate simplifica viața autorilor XML a căror limbă maternă este alta decât limba engleză. Sintaxa DTD ar trebui întotdeauna considerată principală, cu o prescurtare comodă furnizată pentru vorbitorii de limba engleză în forma elementelor predefinite.

Elemente simple

O legătură simplă cu orice nume poate fi definită cu sintaxa următoare în DTD-ul dumneavoastră XML:

```
<!ELEMENT simpleElement ANY>
<!ATTLIST simpleElement
  xlink:type     (simple)        #FIXED "simple"
  xlink:href     CDATA           #REQUIRED
  xlink:role     NMTOKEN         #IMPLIED
  xlink:title    CDATA           #IMPLIED
  xlink:show     (new
                 |replace
                 |embed
                 |undefined)     #IMPLIED
  xlink:actuate  (onLoad
                 |onRequest
                 |undefined)     #IMPLIED >
```

Atributele cele mai semnificative din structura anterioară sunt `show` și `actuate`, care împreună descriu cum se va comporta legătura.

Sugestie de la LeeAnne.com Words to weave by

Singurele lucruri care există efectiv în zona de nume XLink sunt atributele în sine, împreună cu elementele lor predefinite corespunzătoare, deci DTD-ul dumneavoastră trebuie să declare explicit XLink-uri dacă doriți să utilizați propriile dumneavoastră nume.

O legătură echivalentă este predefinită în zona de nume XLink cu numele `simple`. O puteți utiliza direct în codul dumneavoastră dacă declarați zona de nume XLink astfel:

```
<mydocument xmlns:xlink="http://www.w3.org/XML/XLink/" >
  [...]
  <xlink:simple href="URI" actuate="onRequest" attributes >
  [...]
  </xlink:simple>
  [...]
</mydocument>
```

Utilizarea de legături XLink predefinite vă dă o speranță că legătura va fi înțeleasă de agentul dumneavoastră utilizator chiar fără să acceseze DTD-ul documentului și vă scapă de problemele declarării de mecanisme XLink în DTD-ul dumneavoastră – lucru care nu este nici pe departe atât de comod ca utilizarea elementelor predefinite. Un element simplu nu are elemente copil semnificative în zona de nume XLink, deci nu poate fi modificat de alte elemente XLink.

Elemente extinse

Un element extins permite utilizarea tuturor mecanismelor de legare XLink, permițând crearea de legături multidirecționale cu destinații multiple, fie in-line, fie externe:

```
<!ELEMENT extendedElement ANY>
<!ATTLIST extendedElement
  xlink:type     (simple)        #FIXED "simple"
  xlink:role     NMTOKEN         #IMPLIED
  xlink:title    CDATA           #IMPLIED
```



```
  xlink:show     (new
                 |replace
                 |embed
                 |undefined)     #IMPLIED
  xlink:actuate  (onLoad
                 |onRequest
                 |undefined)     #IMPLIED >
```

Elementele extinse pot conține destinații multiple și reprezintă esența XLink, deoarece implementează funcționalitatea extinsă necesară trecerii dincolo de simpla hiperlegătură unidirecțională definită în HTML și XHTML. Un element extins poate avea descendenți XLink semnificativi, cum sunt `locator`, `arc`, `resource` și `title`, adică putem să spunem că acești descendenți *fac* într-adevăr ceva.

Elemente `locator`

Un element `locator` vă permite să identificați punctele terminale ale unui arc de legare XLink:

```
<!ELEMENT locatorElement EMPTY>
<!ATTLIST locatorElement
  xlink:type     (locator)       #FIXED "locator"
  xlink:href     CDATA           #REQUIRED
  xlink:role     NMTOKEN         #IMPLIED
  xlink:title    CDATA           #IMPLIED >
```

Un element `locator` poate avea un descendent XLink `title`.

Elemente `arc`

Un element `arc` descrie arcul de legare efectiv între locații XLink, furnizând reguli de parcurgere care permit agentului utilizator să cunoască în ce direcție indică elementul `arc` și prin ce mecanism de afișare ar trebui afișată resursa de la capătul arcului:

```
<!ELEMENT arcElement EMPTY>
<!ATTLIST arcElement
  xlink:type     (arc)           #FIXED "arc"
  xlink:role     NMTOKEN         #IMPLIED
  xlink:title    CDATA           #IMPLIED
  xlink:show     (new
                 |replace
                 |embed
                 |undefined)     #IMPLIED
  xlink:actuate  (onLoad
                 |onRequest
                 |undefined)     #IMPLIED
  xlink:from     NMTOKEN         #IMPLIED
  xlink:to       NMTOKEN         #IMPLIED>
```

Un element `arc` poate avea un descendent XLink `title`.

Elemente `resource`

Elementele `resource` identifică resursele care participă în legătură, textul vizibil sau alte elemente care pot fi selectate printr-o anumită metodă pentru a activa legătura sau a apela un mecanism de ramificare pentru a alege dintr-o listă de acțiuni posibile.

```
<!ELEMENT resourceElement ANY>
<!ATTLIST resourceElement
  xlink:type    (resource)      #FIXED "resource"
  xlink:role    NMTOKEN         #IMPLIED
  xlink:title   CDATA           #IMPLIED >
```

Un element `resource` nu are descendenți XLink semnificativi.

Elemente `title`

Elementele `title` conțin text lizibil pentru oameni, pe care un agent utilizator sau un browser îl poate prezenta utilizatorului atunci când este nevoie ca acesta să îl înțeleagă mai bine.

```
<!ELEMENT titleElement ANY>
<!ATTLIST titleElement
  xlink:type    (title)         #FIXED "title"
  xlink:title   CDATA           #IMPLIED>
```

Un element `title` nu are descendenți XLink semnificativi.

Din nou despre legăturile simple

O legătură simplă este mai mult sau mai puțin ceea ce face HTML acum, combinând funcțiile lui `locator` și ale legăturii într-un singur element. Deoarece acesta este XML, legătura trebuie să aibă o zonă de nume. Deoarece aceasta este XLink, există mai multe atribute noi care descriu comportamentul legăturii. Numele și zona de nume a atributelor XLink speciale ar trebui să le definiți în DTD, deoarece codul rezultat va fi mai clar. Iată două exemple despre cum ar arăta elementul dumneavoastră XML de legare cu zona de nume XLink declarată în DTD, respectiv fără zona de nume XLink declarată în DTD:

```
<anchor href="pricelist.xml"
  type="simple"
  role="price list"
  title="Price List"
  show="replace"
  actuate="user">Price List</anchor>
```

în comparație cu cel fără declație DTD

```
<anchor xmlns:xlink="http://www.w3.org/XML/XLink/"
  xlink:type="simple"
  xlink:href="pricelist.xml"
  xlink:role="price list"
  xlink:title="Price List"
  xlink:show="replace"
  xlink:actuate="user">Price List</anchor>
```

În ciuda faptului că, în general, XML folosește mai multe cuvinte decât HTML, utilizarea excesivă a zonelor de nume îl face puțin mai greu de utilizat. Profitând de avantajul definirii în elemente a atributelor XLink utilizate de obicei ca legături, puteți face codul rezultat mai ușor de citit, mai ales dacă limba dumneavoastră maternă nu este engleza. Și, evident, aceasta presupune că XML-ul dumneavoastră țintă are acces la DTD-ului documentului și validează față de acesta. În caz contrar, va trebui să utilizați întotdeauna zona de nume XML și să depindeți de posibilitatea ca valoarea zonei de nume XLink să fie implementată ca un comutator în agentul utilizator.

Comparați ambele exemple precedente cu echivalentul aproximativ al etichetei XHTML foarte concise, care are avantajul DTD-ului XHTML, dar nu vă permite să profitați de prea multă flexibilitate, proprie metodei XLink:

```
<a href="pricelist.xml">Price List</a>
```

O altă alternativă, care va combina unele dintre avantajele ambelor metode este să introduceți forțat elementul în zona de nume XLink, astfel:

```
<xlink:a href="pricelist.xml"
  type="simple"
  role="price list"
  title="Price List"
  show="replace"
  actuate="user">Price List</xlink:a>
```

Evident, trebuie să fi declarat anterior zona de nume XLink.

Exemplul următor utilizează o prezentare MathML, pentru o legătură simplă pe care o puteți vedea în Figura 9.1:

```
<math>
 <mrow>
  <mi>y</mi>
   <mo>=</mo>
   <mi></mi>
   <mfrac href="URI" xml:link="simple">
     <mn>1</mn>
     <msqrt>
     <mrow>
     <msup>
      <mi>x</mi>
      <mn>2</mn>
     </msup>
     <mo>+</mo>
     <mn>1</mn>
    </mrow>
   </msqrt>
  </mfrac>
 </mrow>
</math>
```

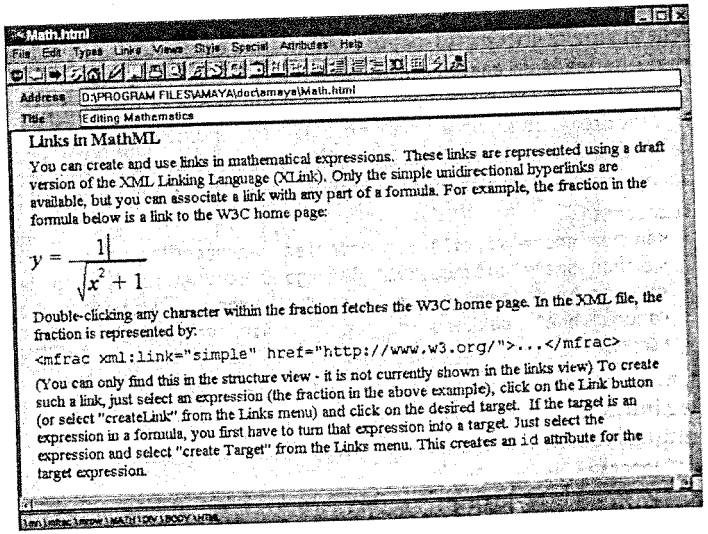


FIGURA 9.1
Există o legătură, dar nu este indicată în ansamblul unei instrucțiuni matematice așa cum se vede în Amaya, browserul W3C.

Din nou despre legăturile extinse

Lucrurile interesante încep o dată cu legăturile extinse. Ele vă permit să plasați, în documente pe care nu le puteți modifica efectiv, legături virtuale de ieșire și alte lucruri interesante. În plus, o legătură extinsă este mult mai ușor de actualizat în cazul în care tipul legăturii se modifică pe durata existenței documentului – de exemplu, pentru a include alte resurse sau pentru parcurgerea legăturii în ambele sensuri.

Exemplul următor creează o legătură bidirecțională într-un document arbitrar:

```
<xlink:extended href="URI"
  type="extended"
  role="price list"
  title="Price List"
  showdefault="replace"
  actuatedefault="user">
Price List
</xlink:extended>
```

Legături in-line

O *legătură in-line* este o legătură în care conținutul său are rolul de element participant la legătură. Mai simplu, o legătură in-line se comportă ca și `<a href="URI">Selection Text</a>` în HTML, unde *Selection Text* activează legătura.

Următorul fragment de cod este un exemplu de legătură in-line:

```
<anchor xlink:href="URI" attributes >Selection Text</anchor>
```

Ca în cazul tuturor legăturilor, attributes ar trebui să conțină valorile `xlink:role=` și `xlink:title=` pentru a asigura lizibilitatea. Deoarece nu toate browserele acceptă aceste atribute la legături, ar trebui ca și `Selection Text` să fie descriptiv. În ciuda faptului că puteți include legături de genul `click here` (clic aici), aceste tipuri de legături furnizează extrem de puține informații pentru utilizatorii cu defecte de vedere.

De asemenea, pe măsură ce tehnologia avansează, acest stil lipsit de imaginație poate împiedica utilizarea include a legăturilor în browserele audio care includ un rezumat al legăturilor la începutul paginii, cum sunt cele folosite în autoturisme. O listă comprimată de legături care include `here` (aici), `click here` (clic aici), `this` (acesta), `here, link` (legătură), `click here, here, here` este puțin probabil să aibă orice fel de utilitate. Faceți o favoare utilizatorilor dumneavoastră incluzând un text descriptiv în textul de selectare. Prin urmare, în listingul următor, care oferă legături la paginile Web ale unor aeroporturi, primul exemplu este mai bun decât cel de-al doilea:

```
<airports>
  <anchor href="sf.xml">San Francisco Airport</anchor>
  <anchor href="chicago.xml">Chicago O'Hare Airport</anchor>
  <anchor href="newyork.xml">New York Kennedy Airport</anchor>
  <anchor href="oakland.xml">Oakland Airport</anchor>
</airports>
```

Cel de-al doilea exemplu utilizează descrieri fără sens și greu de deosebit ale resurselor legăturilor, deci, cu toate că la prima vedere pare să fie disponibilă aceeași informație, în multe browsere ea nu va fi prezentată corect:

```
<airports>
  San Francisco Airport <anchor href="sf.xml">here</anchor>
  Chicago O'Hare Airport <anchor href="chicago.xml">here</anchor>
  New York Kennedy Airport <anchor href="newyork.xml">here</anchor>
  Oakland Airport <anchor href="oakland.xml">here</anchor>
</airports>
```


Utilizați întotdeauna descrieri sugestive ale resurselor legăturii pentru a asigura o flexibilitate maximă în prezentarea legăturii.

Legături externe

În cazul unei *legături externe*, legătura este definită separat de elementele de activare și de localizare. După cum ați învățat anterior în acest capitol, aceasta este cea mai bună variantă. Cu o legătură externă puteți crea legături între două documente care nu au o relație bazată pe autor nici între ele, și nici cu propriile dumneavoastră pagini. Puteți crea legături în documente externe care fac o legătură înapoi la documentul dumneavoastră și viceversa, deci puteți trimite oamenii în locuri străine măcar cu promisiunea unui bilet de întoarcere.

În continuare este prezentată o legătură unidirecțională între un articol de catalog ipotetic și o listă de prețuri asociată. Ambele capete ale legăturii sunt externe pentru documentul care definește legăturile, deși, în general, acest lucru nu este necesar:

```
<xlink:extended
  type="extended"
  role="price list"
  title="Price List"
  showdefault="replace"
  actuatedefault="user">
  <xlink:locator href="URL#lpl142" id="lci9872" />
  <xlink:locator href="URL#lci9872" id="lpl142" />
  <xlink:arc from="lci9872" to="lpl142" show="parsed"/>
</xlink:extended>
```

Dacă doreați o legătură bidirecțională, adăugați pur și simplu încă un arc, astfel:

```
<xlink:arc from="lpl142" to="lci9872" show="parsed"/>
```

Este uimitor de simplu. Figura 9.2 prezintă relația dintre documente. Observați că nu există relație directă între documentul A, din care este făcută legătura, și cele două documente B și C în care este creată legătura. Legătura este vizibilă numai atunci când documentele sunt accesate din documentul A. Un alt document poate crea legături asemănătoare fără a le incomoda cu ceva pe celelalte. Singura cerință este ca atributele `ID` având numele prezentate în exemplu să existe în document. În caz contrar, trebuie furnizată o alternativă de navigare.

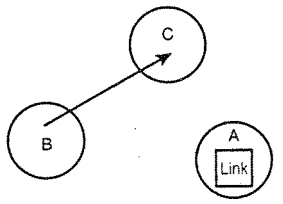


FIGURA 9.2
Iată un XLink unidirecțional dintr-un document extern.

Comportamentul legăturii

Comportamentul legăturii este, în esență, ceea ce se întâmplă atunci când urmăriți o legătură. În prezent există trei acțiuni, `replace`, `new` și `parse`, care înlocuiesc conținutul ferestrei curente, creează o fereastră nouă, respectiv analizează fișierul extern redându-l în fereastra curentă ca și când ar face parte din documentul de bază.

Crearea de ferestre noi nu ar trebui să fie niciodată ceva de la sine înțeles. Acest lucru poate încurca persoanele cu handicapuri vizuale sau cognitive și poate consuma suficiente resurse astfel încât să provoace căderea unor calculatoare cu performanțe reduse. Dacă oamenii doresc să părăsească situl dumneavoastră, lăsați-i. Nu veți câștiga prea mulți prieteni (sau clienți) ținându-i de glezne atunci când încearcă să se târască afară pe ușă. Contrar unei judecăți mai bune, codul următor prezintă o legătură extinsă care afișează o fereastră nouă:

```
<xlink:extended
  type="extended"
  role="price list"
  title="Price List"
  showdefault="replace"
  actuatedefault="user">
  <xlink:locator href="URL#lpl142" id="lci9872" />
  <xlink:locator href="URL#lci9872" id="lpl142" />
  <xlink:arc from="lpl142" to="lci9872" show="new"/>
</xlink:extended>
```

XLink, concluzii

Există multe metode de a utiliza posibilitățile oferite de XLink pentru a realiza legături la alte documente, in-line sau externe, simple sau extinse. Deși sintaxa `simple` are unele avantaje din punct de vedere economic, nu există prea multe motive pentru a o utiliza în practică. Are probleme de compatibilitate cu browserele existente și dă naștere la două opțiuni de legare, iar avantajele nu reușesc să compenseze complexitatea suplimentară.

Notă Un domeniu evident de dezvoltări ulterioare în XLink poate fi asigurarea unui suport clar pentru cadre și, în același timp, conversia lor în ceva rezonabil. Ele nu vor dispărea prea curând din XHTML, deși comportamentul lor poate fi copiat exact prin utilizarea lui XLink fie cu CSS, fie cu foi de stil XSL. Totuși, utilizarea cadrelor este puternic combătută.

XLink permite proiectanților Web să insereze elemente într-o pagină XML pentru a crea și a documenta legături între documente, inclusiv legături între resurse conținute în documente aflate la distanță, fără a fi nevoie de acces de citire la nici unul dintre acestea. Secțiunea următoare studiază XPointer.

XPointer

XPointer furnizează metode extinse de a indica locații specifice sau domenii (*intervale*) din documente. El utilizează ca bază limbajul XPath, care este destul de apropiat de limbajul pointer TEI, care este, la rândul său, înrudit cu limbajul pointer HyTime. După cum s-a prezentat în capitolul anterior, XPath oferă celui care aspiră să creeze legături o sintaxă extinsă și expresivă pentru a descrie locații exacte în cadrul documentelor în funcție de structura documentului sau de tradiționalele ID-uri și ancore.

Notă Pointerii TEI utilizează o notație cu puncte pentru a separa nodurile, și nu notația cu slash folosită de XLink. În afară de aceasta, cele două limbaje de legare sunt similare.

De obicei, XPointer-ii apar ca fragmente URI (numite uneori *selectori*), adică șirul care urmează după # la sfârșitul URI-ului. Există trei metode de a scrie un XPointer:

- Ca prescurtare *idname* simplă pentru `xpointer(id("idname"))`, o facilitate având scopul de a asigura o oarecare compatibilitate cu versiunile anterioare HTML
- Ca serie de numere delimitate prin „/", numite uneori *adrese pas cu pas* sau *adrese cu salturi*
- Ca formă completă, care se poate baza pe XPointer sau pe o altă schemă de pointeri cum este SMIL și poate utiliza orice formă abreviată definită în acea schemă

BIBL. CENTR. UNIV. MIHAIL KOGĂLNICEANU IAȘI

Observați că sintaxa *idname* simplă nu se referă la utilizarea foarte obișnuită a atributului `name=` într-o ancoră HTML `<a>`, cu toate că valorile `id=` și `name=`, în conformitate cu standardul HTML, ar trebui să fie identice dacă ambele atribute sunt prezente într-o ancoră HTML. Deoarece este așa, adăugarea valorilor `id=` corespunzătoare la modelul `name=` mai vechi ar trebui să fie comună atunci când se face conversia de la HTML la XHTML.

Folosirea *idname* este stimulată de această sintaxă deoarece este, probabil, cea mai stabilă formă de referire în documente. Aceasta are mai multe corolare: Deoarece XML poate transclude într-un document părți din alte documente, valorile `id=` ar trebui, dacă este posibil, să fie unice nu numai în documente, dar și pentru mai multe documente. Deoarece este foarte probabil ca propriile dumneavoastră documente să fie legate la altele, cea mai bună metodă de a vă asigura că legăturile la documentele dumneavoastră sunt stabile este să utilizați frecvent și cu atenție valori `id=`. Nu ar fi nepotrivită nici utilizarea unui câmp `id=` în fiecare titlu, sau poate chiar în fiecare paragraf. Dacă prin editări ulterioare se modifică ordinea expunerii dumneavoastră, valorile `id` asigură faptul că referința în sine își păstrează utilitatea chiar și atunci când nu știți cu exactitate cine, cum și unde s-a legat la documentul dumneavoastră.

Sugestie de la LeeAnne.com Words to weave by

Secvențele cu salturi sunt doar serii numerotate de întregi separați prin caractere slash. Fiecare întreg reprezintă al n-lea element copil al elementului anterior. Deși cuvântul „tumbler" (adresă cu salturi) a dispărut din specificații, este încă utilizat. Secvențele cu salturi pot fi utilizate din rădăcină sau din orice nod element dintr-un document.

Cu toate că adresele cu salturi pot fi tentante, deoarece sunt doar serii de selecții de descendenți pornind de la rădăcină în arborele documentului, ele sunt și cele mai sensibile. Dacă vă hotărâți să uniți primele două paragrafe din documentul dumneavoastră în unul singur, atunci fiecare legătură la restul documentului este instantaneu alterată.

Acesta este un exemplu nepotrivit de XPointer utilizând adrese cu salturi:

```
<xlinklocator href="URL#xpointer(/2/3/2/7)" attributes >
```

Acesta este un exemplu mai bun, care utilizează nume de elemente:

```
<xlinklocator href="URL#xpointer(/chap[3]/para[2])" attributes >
```

Acesta este cel mai bun exemplu, folosind un `id` XPointer:

```
<xlinklocator href="URL#lpl2341" attributes >
```

Deşi fiecare XPointer din exemplele anterioare poate adresa aceeaşi locaţie, numai ultimul este probabil să supravieţuiască unei editări extinse. Primul este foarte fragil deoarece adăugarea sau ştergerea unui singur element oriunde în cale va modifica referinţa, cel de-al doilea este mai puţin fragil, dar nu este ideal, deoarece inserarea sau ştergerea unui element chap în oricare din primele trei poziţii va modifica referinţa, asemenea adăugării sau ştergerii unui element para de pe oricare din primele două poziţii.

Secţiunile următoare cercetează modul de utilizare a legăturilor pentru a extinde conceptul de hiperlegături la documente numai pentru citire, începând cu cea mai obişnuită, o legătură unidirecţională la un astfel de document. Vom extinde apoi această funcţionalitate fundamentală cu metode mai complexe care permit legarea *de la* documente numai pentru citire şi legături multiple.

Realizarea de legături la documente numai pentru citire

Nu există diferenţe între crearea unei legături extinse la un document numai pentru citire şi orice alt document. Pentru fiecare se foloseşte aceeaşi sintaxă, singura excepţie fiind aceea că baza legăturii nu poate fi în documentul numai pentru citire, iar pentru păstrarea legăturii trebuie să utilizaţi o pagină de referire externă. Exemplul următor creează o legătură bidirecţională între două documente, în care puteţi presupune că nu aveţi acces de scriere:

```
<xlink:extended
  type="extended"
  role="price list"
  title="Price List"
  showdefault="replace"
  actuatedefault="user">
  <xlink:locator href="URL#lpl142" role="lci9872" />
  <xlink:locator href="URL#lci9872" role="lpl142" />
  <xlink:arc from="lci9872" to="lpl142" show="new"/>
  <xlink:arc from="lpl142" to="lci9872" show="new"/>
</xlink:extended>
```

Numele role din referinţele de mai sus sunt, în mod deliberat, arbitrare, deoarece nu trebuie să ştim efectiv ce relaţie există între documente. Referinţele la direcţiile role şi arc pot fi, de fapt, generate şi inserate prin programare în documentul sursă.

Adresare relativă

Exemplele anterioare au prezentat adresarea absolută printr-un ID conţinut în fiecare document, lucru care poate fi făcut acum cu elemente ancoră HTML simple. Dar întreaga sintaxă XPath plus extensiile XPointer sunt disponibile pentru navigarea prin document. După cum aţi învăţat anterior în acest capitol, optaţi întotdeauna pentru a naviga, dacă este posibil, de la un ID intern cât mai apropiat cu putinţă de ţinta dumneavoastră. Exemplul următor prezintă o navigare pur relativă de la rădăcină, de la cel de-al treilea copil al celui de-al optulea copil al rădăcinii şi de la cel de-al cincilea copil al celui de-al patrulea copil al rădăcinii. Fiecare locaţie are un ID, deci ambele capete ale arcului XLink care le uneşte pot fi localizate cu uşurinţă, dar navigarea rezultată este foarte nesigură deoarece orice modificare a numerelor de elemente dinaintea elementelor selecţiei modifică în mod irevocabil locaţia:

```
<xlink:extended
  role="price list"
  title="Price List"
  showdefault="replace"
  actuatedefault="user">
  <xlink:locator href="URL#xpointer(/8/3)" id="lci9872" />
  <xlink:locator href="URL#xpointer(/5/4)" id="lpl142" />
  <xlink:arc from="lci9872" to="lpl142" show="new"/>
  <xlink:arc from="lpl142" to="lci9872" show="new"/>
</xlink:extended>
```


Ambele fragmente de URI adresează elemente copil fără nume, relative la rădăcină. Observaţi că ambele capete ale arcului sunt relativ nesigure, deşi ele însele au valori ID pentru a uşura referirea prin elemente arc, deoarece documentul poate fi editat astfel încât locaţia dorită să nu mai fie referită de expresia de mai sus.

Adresare absolută

Aţi văzut deja adresarea absolută care depinde de găsirea unui element având un ID tocmai în locul unde doriţi să ajungeţi. În documentele dumneavoastră puteţi ajuta acest proces pentru propriile dumneavoastră legături, precum şi pentru acelea care pot fi făcute de alţii, prin includerea de ID-uri unice pentru fiecare element semnificativ. Iată un alt exemplu care utilizează adresarea absolută:

```
<xlink:extended
  role="price list"
  title="Price List"
  showdefault="replace"
  actuatedefault="user">
  <xlink:locator href="URL#lpl142" id="lci9872" />
  <xlink:locator href="URL#lci9872" id="lpl142" />
  <xlink:arc from="lci9872" to="lpl142" show="new"/>
  <xlink:arc from="lpl142" to="lci9872" show="new"/>
</xlink:extended>
```

Ambele capete ale arcului din acest exemplu utilizează ID-uri fără adresare relativă de la acea bază. Aceasta este cea mai stabilă formă de adresare, deoarece autorii ştiu că o locaţie cu un ID este probabil să fie referită din exteriorul documentului şi vor evita în mod raţional modificarea sau ştergerea ID-ului atunci când editează sau actualizează documentul.

Selectare după nume

Selectarea după numele elementului este mai puţin puternică decât selectarea după ID, dar mult mai bună decât simpla adresare relativă. Chiar şi cea mai mică modificare dintr-un document poate altera o cale relativă, în timp ce modificarea unui element poate să afecteze sau să nu afecteze o selecţie bazată pe numele elementului, după cum modificarea afectează sau nu elemente având acelaşi nume cu selecţia. Este posibil chiar să combinaţi cele două stiluri, utilizând adresarea prin ID pentru a găsi un loc lângă destinaţia dumneavoastră finală şi navigând din acel loc în funcţie de numele elementului. Exemplul următor utilizează această sintaxă cu nume de element:

```
<xlink:extended
  type="extended"
  role="price list"
  title="Price List"
  showdefault="replace"
```

```
  actuatedefault="user">
  <xlink:locator href="URL#xpointer(/chap[8]/para[3])" id="lci9872" />
  <xlink:locator href="URL#xpointer(id("lcp21")/para[3])" id="lpl142" />
  <xlink:arc from="lci9872" to="lpl142" show="new"/>
  <xlink:arc from="lpl142" to="lci9872" show="new"/>
</xlink:extended>
```

Exemplul precedent utilizează atât o serie de nume de elemente pornind de la rădăcină, cât și un singur nume de element de la un element identificat printr-un ID. Nici una nu este pe deplin sigură, dar sunt mult mai sigure în cazul unei simple editări la o locație anterioară decât ar fi o simplă adresă relativă. De exemplu, scrierea cu aldine a primului cuvânt al unui capitol va modifica numărul de elemente copil, dar nu și numărul de capitole sau paragrafe.

Extensii XPointer

XPointer utilizează toate funcțiile definite în XPath, dar cu foarte multe extensii pentru a le îmbunătăți aplicabilitatea la operația primară din XPointer, adresarea într-o legătură a părților unui document. În ciuda faptului că simplele construcții XPath oferă destul de multă putere, XPointer a făcut, la rândul său, mai multe extensii la setul XPath pentru a extinde utilizările posibile. Deși construcțiile următoare par asemănătoare cu elementele `locator` pentru noduri, ele nu pot fi utilizate decât ca unic fragment inițial de element `locator` dintr-un URI și returnează valori care nu sunt noduri și chiar pot să nu conțină noduri. Este încă destul de neclar cum vor folosi această informație agenții utilizator:

- `range::`*`XPath-node-locator`* `to` *`XPath-node-locator`* – Preia ca argumente o pereche de elemente `locator` pentru noduri XPath separate de „to" și returnează agentului utilizator șirul care conține prima locație și continuă până la ultima locație, inclusiv orice marcaj conținut. Iată un exemplu:

  ```
  xpointer(id("sec2.1")/descendant::P[last()] to
           id("sec2.2")/descendant::P[last()])
  ```

  Deoarece XPointer vă permite și selectarea până la nivel de caracter, puteți să selectați orice porțiune contiguă a unui document XML.
- `string-range::"`*`literal`*`"?,`*`integerposition`*`?,`*`integerlength`*`?,`*`predicate`*`?` – Preia o listă separată prin virgule, formată dintr-un șir `literal` încadrat între ghilimele, urmat de un număr `integerposition`, un număr `integerlength` și un `predicate` XPath opțional, și caută doar în textul documentului, ignorând marcajele și unind spațiile albe într-un singur spațiu pentru `literal` dacă acesta este inclus, iar în caz contrar operează de la caracterul de dinainte de `integerposition` și returnează `integerlength` caractere ale instanței selectate prin `predicate`, dacă acesta este prezent.

Pot apărea mai multe potriviri, dar acestea nu se suprapun. Deci, `string-range::"hopa",,9,[position()=2]` acționând asupra textului `"hopa hopa hopa hopa"` returnează un șir conținând ultimele două șiruri `hopa` într-un singur șir, `"hopa hopa"`, și nu cele două din mijloc.

Un `integerposition` omis se presupune că este 1 și nu poate avea valoarea 0. Dacă este negativ, numărătoarea se face de la sfârșitul șirului spre început, în timp ce dacă este pozitiv numărătoarea se face de la poziția dinaintea primului caracter selectat de `literal` spre sfârșitul șirului. Nu este o eroare dacă `integerposition` se află în afara șirului în sine.

Un `integerlength` omis se presupune că este zero, iar valorile negative nu sunt permise, deci `integerlength` are întotdeauna o direcție pozitivă. Nu este o eroare dacă lungimea trece de capătul documentului, dar în această situație este returnat numai șirul până la sfârșitul documentului.

Funcții XPointer pentru locații absolute

Funcțiile pentru locații absolute sunt pur și simplu acelea care returnează o locație efectivă, independentă de context. Deci funcția `id()` este independentă de context deoarece un ID trebuie să fie unic în document, iar funcția returnează o locație sau nici una.

- *`location-set`* `id(`*`IDliteral`*`)` – Funcția `id(`*`IDliteral`*`)` returnează locația nodului care conține ID-ul `IDliteral`.
- *`location-set`* `here()` – Funcția `here()` returnează locația nodului in-line a lui XPointer din documentul în care se găsește cursorul documentului. Nu preia argumente.

Calea rădăcină, `/`, o cale locație absolută în XPointer, este localizată înaintea oricărui marcaj din document, după cum este descris în Capitolul 8, „XPath". Deci nu va fi returnată niciodată de `here()` ca locație.

- *`location-set`* `origin()` – Funcția `origin()` returnează locația nodului extern al punctului în care un utilizator a pornit o parcurgere de noduri. Nu preia argumente.
- *`location-set`* `start-point(`*`location-set`*`)` – Funcția `start-point(`*`location-set`*`)` returnează punctul de pornire al fiecărei locații din argumentul `location-set`.
- *`location-set`* `end-point(`*`location-set`*`)` – Funcția `end-point(`*`location-set`*`)` returnează punctul de sfârșit al fiecărei locații din argumentul `location-set`.

Funcții booleene XPointer

Există o singură extensie la funcțiile booleene XPath, o metodă prescurtată de a spune dacă o operație a returnat un singur nod, economisind astfel o comparație. Funcția este `boolean unique()`. Funcția `unique()` returnează valoarea adevărat dacă și numai dacă lista de noduri context conține un singur nod.

Selectarea unui interval

Selectarea unui interval necesită utilizarea sintaxei extinse pentru intervale din XPointer:

```
<xlink:simple
  type="simple"
  role="price list"
  title="Price List"
  show="parse"
  actuate="user"
  href='URL#xpointer(id("lap1248") to id("lap28")'>Select this link for Price
       ➥List</xlink:simple>
```

Selectarea șirurilor

Utilizarea unui interval într-un șir necesită folosirea extensiei XPointer `string-range`:

```
<xlink:simple
  href='URI#xpointer(id("lap234")/string-range(/,"text",1,10)'
  type="simple"
  role="code"
  title="Code"
  showdefault="replace"
  actuatedefault="user">
</xlink:simple>
```

Presupunând că aceste coduri au fost identificate prin ID, acesta va returna un șir de 10 caractere conținând codul.

Depanarea XLink și XPointer

Erorile posibile în XLink și XPointer nu sunt chiar atât de multe. Deoarece ele pot fi verificate în timpul creării și ar fi destul de simplu să se adauge la un editor grafic de documente XML capacitatea de a selecta vizual o locație sau un interval, va fi probabil destul de simplu să vă asigurați că legăturile funcționează chiar de la început. Dar multe pagini se modifică zilnic, deci ingeniozitatea celui mai inteligent proiectant va fi greu încercată pentru găsirea unui punct de referință stabil sau a unei structuri uniforme.

Manipularea legăturilor întrerupte

Pe neașteptate, după legarea la un document, observați că legăturile dumneavoastră nu mai funcționează.

Legăturile sunt printre cele mai fragile realizări umane de pe Web. Totuși, cu XLink există chiar mai multe șanse de a face erori, deoarece este posibil să depindeți nu numai de adrese URL, ci și de fragmente URI care identifică o locație din document. Totuși, există strategii care minimizează șansele ca o legătură să o ia razna, deși aceasta depinde de inspectarea atentă a documentului sursă. În primul rând, legăturile relative care pornesc de la rădăcină sunt mai delicate decât legăturile care navighează pornind de la repere interne. Aproape orice modificare din document poate schimba calea suficient pentru a trimite utilizatorii dumneavoastră aiurea.

Legăturile care utilizează un ID intern sunt de obicei mai robuste, deoarece autorii de documente încearcă să păstreze ID-urile, pentru că propriile lor legături pot depinde de acestea. Deci găsirea unui punct ancoră existent lângă punctul de pornire selectat de dumneavoastră poate ajuta la menținerea funcționării corecte a legăturii.

Dacă documentul nu are puncte ancoră interne, atunci încercați să identificați structura de bază, titluri, secțiuni, orice are sens în documentul în sine. În loc să spuneți că legătura dumneavoastră indică la cel de-al 144-lea element copil al rădăcinii, vedeți dacă identificarea sa ca al doilea paragraf din secțiunea 8 nu creează o legătură mai stabilă în timp.

Identificarea tipului elementului de la destinația dumneavoastră face, de asemenea, calea mult mai robustă, deoarece ocolește modificările în alte tipuri de elemente. Alegeți întotdeauna, dacă este posibil, tipuri de elemente cu nume în locul relațiilor copil sau frate.

Verificarea sintaxei legăturilor

După codarea unui document utilizând XLink, veți dori să vă asigurați că legăturile funcționează corect, dar browserul dumneavoastră favorit nu acceptă încă XLink.

Pentru a testa legăturile, trebuie să utilizați un instrument care suportă XLink-uri. InDelv este o opțiune. Sau analizați chiar dumneavoastră sursa cu un analizor care înțelege XLink-urile. Dacă XLink-urile dumneavoastră pot fi modificate ca să respecte sintaxa acceptată, există un verificator de legături la pagina de bază a lui W. Eliot (Dr. Macro) Kimber, la: `http://www.drmacro.com/hyprlink/xlink/xptrtest.html`. Acest instrument are anumite limitări de sintaxă, așa cum este descris chiar în pagina respectivă.

Partea a II-a

Manipularea XML

10 DOM (Document Object Model) – Modelul document obiect
11 SAX (the Simple API for XML) – API simplu pentru XML
12 CSS1, CSS2, DSSSL, XSL
13 Foi de stil în cascadă și XML/XHTML
14 Utilizarea generației următoare de foi de stil extensibile – XSL

CAPITOLUL 10

DOM (Document Object Model) – Modelul document obiect

În acest capitol

DOM
Identificarea diferențelor între DOM Level 1 și DOM Level 2
Un copac poate ascunde o pădure
Utilizarea DOM la nivel de nod
Utilizarea unui DOM orientat spre obiecte
Structura DOM
Studiul capriciilor din DOM
Tratarea excepțiilor
Studii de caz

DOM

Modelul document obiect (DOM) este o metodă de a realiza o interfață cu un document XML în termenii structurii sale de bază, utilizând o interfață de programare a aplicațiilor (API – Application Programming Interface). DOM este interesant numai dacă sunteți pregătit să programați pentru API utilizând măcar un limbaj de scriptare, deși majoritatea lucrurilor actuale par să fie făcute în Java sau în alte limbaje orientate spre obiecte. Dar dacă doriți să faceți puțină programare, accesarea DOM-ului vă oferă control total asupra documentului, permițându-vă să căutați în el și să îl modificați în aproape orice mod pe care vi-l puteți imagina.

Obiectele DOM sunt accesate prin *metode*, un mod de comunicare cu obiectele, definit precis, care este cercetat mai amănunțit în secțiunea din acest capitol dedicată metodelor. Metodele sunt fundamentale pentru o perspectivă orientată spre obiecte asupra lumii; dacă un API nu ascunde informații și obligă interacțiunea prin metode, atunci nu este pe deplin orientat spre obiecte. DOM-ul XML îndeplinește ambele condiții.

DOM ascunde complet structura fizică de bază a unui document XML. În ciuda faptului că, din punct de vedere intern, poate fi considerat ca fiind un arbore, un document XML este stocat de cele mai multe ori ca fișier secvențial plat. Totuși, nu există nimic care să împiedice stocarea unui document XML sub forma unei baze de date relaționale

sau folosind orice altă tehnologie de stocare care vă vine în minte, atât timp cât este scrisă o interfață care să prezinte lumii exterioare structurile și metodele DOM.

DOM obligă accesul prin metode. Deși vi se pare că puteți accesa întregul document, dumneavoastră accesați de fapt numai reprezentarea sa. Etichetele efective utilizate într-un fișier secvențial plat sunt invizibile pentru DOM, iar pointerii și interogările necesare pentru a accesa o reprezentare echivalentă a unui document XML sub forma unei baze de date relaționale sunt de asemenea invizibile.

DOM definește două niveluri de structură: o vedere complet orientată spre obiecte a documentului, ca un obiect sau o colecție de obiecte (după cum spune și numele specificației), și o vedere la nivel de nod a cărei prelucrare este mai puțin costisitoare, dar totuși orientată spre obiecte, deoarece și interfața mai simplă este accesată tot prin metode.

În aplicațiile mici, cheltuielile de prelucrare pentru crearea și utilizarea vederii complet orientate spre obiecte sunt prea mari pentru ca aceasta să fie practică. Având o metodă alternativă de adresare a documentului înseamnă că este posibilă efectuarea de prelucrări semnificative ale documentului și pe dispozitive mici, poate chiar hand-held.

Notă

Pentru a simplifica foarte mult, *orientat spre obiecte* este un termen tehnic pentru o tehnologie de programare care permite modulelor program să se comporte ca și când ar fi, într-un fel, vii, sau măcar entități separate. Un obiect este capabil să se inițializeze pe el însuși, o potențială sursă de erori în modelele mai vechi, și să moștenească însușiri de la „părintele" său. Asemenea unei ființe vii, un obiect are un interior și un exterior. Interiorul este mai mult sau mai puțin ascuns privirii. Accesați și manipulați obiectul comunicând cu el printr-un vocabular foarte controlat, format din *metode*, în același fel în care puteți învăța un câine să facă un lucru simplu, spunându-i „Șezi!". Atunci când spuneți că ceva este orientat spre obiecte, spuneți că are proprietăți care vă permit să îl tratați ca pe o cutie neagră.

Vederea la nivel de nod are ceva în comun cu SAX (the Simple API for XML – API simplu pentru XML), discutat în capitolul următor, și cu alte analizoare XML conduse prin evenimente, deoarece toate lucrează cu noduri și, din punct de vedere logic, își realizează funcțiile prin parcurgerea arborelui documentului. Totuși, spre deosebire de analizoarele conduse prin evenimente, DOM vă permite să parcurgeți arborele așa cum doriți. De exemplu, puteți parcurge arborele în preordine sau în postordine, sau să interogați și să modificați documentul pe loc. Acest lucru ar fi posibil în SAX numai prin treceri multiple prin document și este studiat în Capitolul 11, „SAX (The Simple API for XML) – API simplu pentru XML".

Notă

În momentul actual, XML nu dispune de nici un mecanism pentru asocierea evenimentelor dinamice cu un document, în afară de lucrul în zona de nume XHTML. DOM Level 2 conține un astfel de instrument, care ar trebui să fie disponibil în momentul în care citiți aceste rânduri sau puțin după aceea. O altă posibilitate de a include comportament dinamic într-un document este extinderea CSS cu extensii comportamentale care să permită script-uri, sau ceva echivalent, care să fie apelate dintr-o foaie de stil CSS de nivelul 3 sau superior, dar acest lucru este într-o fază incipientă la W3C.

De fapt, atunci când priviți o interfață la nivel de nod, observați că seamănă puțin cu celălalt standard elementar XML la nivel de nod, XPath. Vedeți aceleași relații părinte, copil și frate, deși DOM utilizează o sintaxă complet diferită.

Ar trebui reținut că DOM este un API, și nu o descriere a structurii documentului. Cu toate că de multe ori este mai comod să privim un document XML ca pe un arbore, DOM în sine nu face asemenea presupuneri despre reprezentarea de bază a unui document XML. Acest lucru este logic în cazul unei abordări orientate spre obiecte; nu există nici un motiv ca cineva să știe cum este stocat documentul. DOM-ul îl puteți implementa pe un document bază de date care utilizează tehnici de prelucrare relaționale, de tip fișier plat, arbore, orientate spre obiecte sau chiar asociative, iar rezultatul va fi, prin definiție, exact același.

Dynamic HTML în Netscape și MSIE

Prelucrarea Dynamic HTML este ceea ce a dus la DOM, deoarece atât Netscape cât și Microsoft au apărut cu modele obiect ale documentului HTML pentru a realiza propriile lor scamatorii brevetate asupra paginilor HTML. Ambele au fost supuse spre examinare grupului W3C la momentul potrivit, iar W3C a fost impresionat mai mult de ideea Microsoft. Deci Netscape a fost lipsit de suportul DOM și a încercat să evite problema prin concentrarea asupra lucrurilor inteligente care pot fi făcute cu niveluri (layer). Dar DOM W3C permite accesul la întregul document, în timp ce versiunea Netscape este indiscutabil mai limitată.

Având acest avantaj, Microsoft a beneficiat de pe urma incipientei compatibilități cu DOM-ul W3C, în timp ce Netscape a rămas cu mult în urmă în privința suportului DOM. La apariția Mozilla, Netscape a promis că va fi complet compatibil DOM, poate chiar mai mult decât versiunea Microsoft, deoarece ei nu sunt interesați de moștenirea completă a compatibilității parțiale și a extensiilor brevetate.

Și Microsoft are propriile probleme cu DOM, deoarece ceea ce a standardizat W3C nu era chiar ceea ce propusese Microsoft. Deci DOM-ul Microsoft este presărat cu capcane pentru dezvoltările brevetate imprudente care fac ca paginile codate cu versiunea Microsoft de DOM să se întrerupă în oricare alt browser. Calea spre standardizarea completă de la stadiul actual al ambilor mari furnizori de browsere este dificilă și plină de riscuri din cauza numărului imens de utilizatori. Din fericire, majoritatea oamenilor nu au folosit DOM chiar atât de mult.

Lipsurile DOM

Din unele nemulțumiri, vă puteți închipui că totul lipsește sau este inadecvat, dar DOM Level 1 a fost destinat de la început să fie înlăturat și probabil că așa se va întâmpla până în momentul în care veți citi aceste rânduri. DOM Level 2 este acum un candidat de Recomandare, aflat înaintea transformării sale în Propunere de Recomandare și apoi într-o Recomandare completă. Desigur, DOM Netscape 4.x este complet străin și va fi înlocuit complet în Mozilla.

DOM a fost proiectat să suporte HTML, iar DOM Level 1 este de fapt o unealtă HTML. Cerințele minime pentru HTML formează încă miezul lui DOM Level 2, dar au fost adăugate multe lucruri pentru suportul XML și în special pentru standardele legate de acesta. Deoarece DOM Level 2 este necesar pentru a suporta cu adevărat XML, el este prezentat aici ca și când ar fi deja un standard, pentru că sunt deja disponibile seturi de instrumente care permit experimentarea cu interfața – chiar dacă nu este complet compatibil cu un standard care încă suferă modificări.

Identificarea diferențelor între DOM Level 1 și DOM Level 2

DOM Level 1 existent nu suportă toate caracteristicile din actualul XML 1.0. Multe caracteristici necesare pentru suportul CSS, în special caracteristicile CSS2, nu există, și pe lângă acestea mai lipsesc multe din comportamentele specifice XML. În acest capitol, DOM este tratat ca un tot unitar, menționând de la început diferențele pentru ca dumneavoastră să știți ce teme este posibil să nu fie implementate complet până la apariția acestei cărți.

Sugestie de la LeeAnne.com Words to weave by

Sunteți încurajat să utilizați marginile acestei cărți ca să adăugați propriile dumneavoastră note la această secțiune și la celelalte, pentru a vă reaminti de caracteristicile modificate sau numai pentru a reține propriile dumneavoastră reacții la aceste informații. Paginile unei cărți ar trebui să rămână curate numai cât timp se află în librărie.

Cu toate că la momentul scrierii acestei cărți DOM-2 are statutul de candidat de Recomandare, reacțiile venite de la implementatori pot încă duce la modificarea substanțială a documentului de către grupul de lucru. Aceasta înseamnă că anumite caracteristici menționate pot fi modificate sau șterse, iar caracteristici noi pot fi adăugate, deși direcția generală a DOM-2 este deja clară.

Atribute și metode noi

Acestea reprezintă modificări aduse interfețelor existente pentru a extinde funcționalitatea lor în diverse moduri. În multe situații, ele există pentru a suporta cerințele adăugate de standardele legate de XML, pe măsura apariției lor. Aceste cerințe se pot modifica, probabil în direcția adăugării de caracteristici, înainte de finalizarea Recomandării DOM-2, acum în stadiul de candidat, deoarece multe dintre standardele legate de XML sunt încă în schimbare. Lista următoare conține caracteristicile adăugate până acum la DOM Level 2:

- Interfața `Attr` are un atribut nou: `ownerElement`. Acesta permite accesul ușor în special la elementul proprietar al unui ID, sau la orice atribut, și este o componentă cheie a CSS, XLink, XPath și a altor standarde legate de XML.
- Interfața `Document` are cinci metode noi: `importNode`, `createElementNS`, `createAttributeNS`, `getElementsByTagNameNS` și `getElementByID`. Prima dintre ele suportă XLink, în timp ce următoarele trei suportă zonele de nume XML, iar ultima suportă în mare parte CSS, dar și alte standarde.
- Interfața `NamedNodeMap` are trei metode noi: `getNamedItemNS`, `setNamedItemNS` și `removeNamedItemNS`. Toate suportă zonele de nume XML.
- Interfața `Node` are două metode noi: `supports` și `normalize`. Acestea sunt specifice XML și permit programatorului să determine dacă implementarea DOM suportă o anumită caracteristică și să unește nodurile text adiacente într-unul singur.
- Interfața `Node` are trei atribute noi: `namespaceURI`, `prefix` și `localName`. Toate acestea au rolul de a suporta zonele de nume XML.
- Atributul `ownerDocument` a fost specificat să fie nul atunci când nodul este un document. Acum este nul și dacă nodul este un `DocumentType` care nu este utilizat deocam-

dată cu nici un document. Acesta are rolul de suporta transcluderea XML și alte transformări ale documentelor.

- Interfața `DocumentType` are trei atribute noi: `publicID`, `systemID` și `internalSubset`. Acestea suportă caracteristicile XML asociate declarației documentului XML.
- Interfața `DOMImplementation` are două metode noi: `createDocumentType` și `createDocument`.
- Interfața `Element` are șase metode noi: `getAttributeNS`, `setAttributeNS`, `removeAttributeNS`, `getAttributeNodeNS`, `setAttributeNodeNS` și `setElementsByTagNameNS`. Toate suportă zonele de nume XML.
- `DOMException` are cinci coduri de excepții noi: `INVALID_STATE_ERR`, `SYNTAX_ERR`, `INVALID_MODIFICATION_ERR`, `NAMESPACE_ERR` și `INVALID_ACCESS_ERR`.

Interfețe noi

Interfețele noi, descrise complet în secțiunile următoare, reprezintă funcționalități complet noi adăugate la DOM-2.

HTML

Caracteristicile HTML sunt proiectate să suporte documentele HTML, care diferă de documentele XML generale prin faptul că au un DTD predefinit care poate fi implementat complet în agentul utilizator.

Multe interfețe și atribute sunt numai pentru comoditate și oferă un mod alternativ de accesare a informațiilor disponibile în alte moduri, dar unele sunt proiectate să adreseze elemente HTML specifice. Interfața `HTMLDOMImplementation` a fost adăugată la modulul HTML existent. Ea este proiectată să permită aplicațiilor să determine nivelul de suport al implementării DOM curente, opțional printr-o anumită caracteristică, astfel încât poate fi asigurată independența acelei implementări.

Vederi (Views)

Vederea unui document (*document view*) este una din multiplele prezentări posibile asociate unui document prin intermediul unei foi de stil sau al altui mecanism pentru alegerea unei vederi din mai multe posibile. O vedere poate fi determinată de un tip de mediu, de aptitudinile agentului utilizator sau poate fi chiar modificată dinamic prin interacțiunea utilizatorului. Acest modul nou definește interfețele `AbstractView` și `DocumentView`. Acestea furnizează un mijloc de bază pentru asocierea într-o aplicație a diferitelor vederi ale unui document, fie procesate, fie după aplicarea unei foi de stil.

Foi de stil (style sheets)

Acest modul nou definește interfețele următoare, proiectate să furnizeze interfețele fundamentale necesare pentru a suporta orice limbaj pentru foi de stil, inclusiv CSS, dar nu numai CSS la el. Interfețele `StyleSheet`, `StyleSheetList`, `MediaList`, `DocumentStyle` și `LinkStyle` suportă atât CSS cât și XSL, precum și alte limbaje pentru foi de stil.

CSS (foi de stil în cascadă)

Acest modul nou definește interfețele următoare, proiectate să suporte foile de stil în cascadă, etalând programatorului construcțiile efective ale foilor de stil:

`CSS2Azimuth`	`CSS2BackgroundPosition`	`CSS2BorderSpacing`
`CSS2CounterIncrement`	`CSS2CounterReset`	`CSS2Cursor`
`CSS2FontFaceSrc`	`CSS2FontFaceWidths`	`CSS2PageSize`
`CSS2PlayDuring`	`CSS2Properties`	`CSS2TextShadow`
`CSSCharsetRule`	`CSSFontFaceRule`	`CSSImportRule`
`CSSMediaRule`	`CSSPageRule`	`CSSPrimitiveValue`
`CSSRule`	`CSSRuleList`	`CSSStyleDeclaration`
`CSSStyleRule`	`CSSStyleSheet`	`CSSUnknownRule`
`CSSValue`	`CSSValueList`	`Counter`
`RGBColor`	`Rect`	`ViewCSS`
`DocumentCSS`	`DOMImplementationCSS`	`HTMLElementCSS`

După cum puteți vedea din numele modulului și interfețelor, toate aceste adăugiri au fost făcute pentru a suporta caracteristicile CSS și CSS2. `CSS2Azimuth2`, de exemplu, este necesar pentru foile de stil auditive din CSS2.

Notă Accesarea unei foi de stil a unui utilizator nu este corectă și nu vor fi furnizate mijloace care să dezvăluie deciziile de stil ale utilizatorului.

Evenimente (Events)

Evenimentele sunt asociate unui document în timpul în care acesta este redat sau afișat. Ele furnizează un mijloc concret cu ajutorul căruia o aplicație poate realiza alte prelucrări dinamice decât cele definite prin hiperlegături sau alte mecanisme încorporate în limbaj. Exemplele de evenimente includ acțiuni de la mouse sau de la tastatură, modificarea zonei active sau acțiuni care modifică într-un fel structura documentului. Ele sunt asemănătoare cu evenimentele primitive la nivel de atribut asociate documentelor de HTML, dar mult mai bogate.

În modelul nou, evenimentele pot fi capturate de un strămoș al elementului țintă, modificând prin aceasta comportamentul de interschimbare definit de HTML pentru a permite ca prelucrarea evenimentelor să aibă loc descendent. De asemenea, un eveniment poate fi anulat de orice handler pentru a preveni efectuarea unor acțiuni prestabilite.

Acest modul nou definește următoarele interfețe: `Event`, `EventListener`, `EventTarget`, `MutationEvent`, `UIEvent`, `MouseEvent` și `KeyEvent`. Aceste interfețe noi permit programatorului să acceseze acțiuni ale mouse-ului și alte evenimente interactive asociate unui document XML.

Parcurgere (Traversal)

Caracteristica de parcurgere permite programatorului să parcurgă cu ușurință arborele logic, fie într-o secvență unidimensională plată de noduri în ordinea documentului, fie într-o structură bidimensională completă de arbore. Un `NodeIterator` permite parcurgerea înainte sau înapoi de-a lungul reprezentării plate, în timp ce `TreeWalker` parcurge un arbore bidimensional. Fiecare vedere a arborelui poate fi modificată de filtrele încorporate, numite *indicatoare* `whatToShow`, sau de un filtru personalizat, scris în acel sens.

Filtrele sunt mecanisme foarte puternice pentru simplificarea prelucrării arborelui, deoarece orice detalii neesențiale pot fi îndepărtate cu ușurință din vedere, permițând aplicației să își concentreze prelucrarea numai asupra acelor părți ale documentului care sunt de interes.

Acest modul nou definește următoarele interfețe: `NodeIterator`, `TreeWalker`, `NodeFilter` și `DocumentTraversal`. Aceste interfețe noi definesc metode de traversare a arborelui documentului.

Interval (Range)

Un interval este o submulțime continuă a unui document, mărginită de o pereche de locații limită. Din cauza prelucrării de către un agent utilizator sau prin alte metode, un interval poate să nu pară continuu atunci când este afișat.

Acest modul nou definește două interfețe noi: `Range` și `DocumentRange`, având asociată și o excepție, `RangeException`. Aceasta a fost adăugată pentru a suporta standardele XLink, permițându-vă să specificați prin intervale subarborii unui document XML.

Diferențele dintre DOM-1 și DOM-2, concluzii

După cum puteți vedea, majoritatea modificărilor implică suportul pentru interactivitate, foi de stil sau cerințe XML. Dar există câteva noutăți interesante care afectează în profunzime fiecare document, în special noile interfețe de parcurgere care permit construirea de vederi arbore diferite ale arborelui documentului utilizând excluderi de noduri, care corespund în mare unui plan de crâng (grove) în SGML/HyTime/DSSSL. Aceasta vă permite să reduceți arborele documentului astfel încât el să corespundă scopurilor dumneavoastră, ignorând etichetele pe care nu vreți să le redați.

Notă Multe dintre modulele noi sunt opționale. Înainte de a le utiliza trebuie să testați dacă versiunea dumneavoastră DOM le suportă sau nu, utilizând fie metoda `supports()` din DOM Level 2, fie metoda `hasFeature()` din DOM Level 1 și 2.

Lipsesc încă metodele unitare de instanțiere a obiectelor DOM, controalele de acces și de securitate, accesul la DTD-ul XML și validările. Schița de lucru pentru DOM Level 3 trebuia să apară în aprilie 2000, deci ar trebui să știți deja ce a plănuit W3C în momentul apariției acestei cărți sau puțin mai târziu.

Un copac poate ascunde o pădure

Deoarece XML vă permite să realizați legături la alte documente cu ajutorul instrucțiunilor de prelucrare, al notațiilor și atributelor care nu sunt copiii nodului care le conține (un oximoron nefericit), sau să transcludeți dinamic documente din afara documentului de bază, există mulți arbori logici care pot fi construiți din punct de vedere teoretic având ca bază același document. În plus, este posibil să nu fiți nevoit să vizualizați sau să redați, în multe aplicații, întregul arbore, definind un alt arbore logic alcătuit dintr-o submulțime a arborelui complet al documentului XML. Fără a exagera, pot exista sute de arbori care au rădăcinile într-un singur document. Pentru comoditate, colecția tuturor arborilor este numită uneori pădure sau crâng.

Deși momentan nu există o metodă simplă pentru manipularea crângurilor, DOM Level 2 extinde specificația DOM 1 cu instrumente de parcurgere care fac deplasarea prin crânguri simplă și ușoară. DOM Level 2 și următoarele variante vor furniza în cele din urmă o vedere clară a pădurii prin copaci, permițându-vă aproape orice perspectivă pe care v-o puteți imagina asupra documentului. Instrumentele de bază pentru manipularea vederii documentului sunt indicatoarele `whatToShow` și filtrele scrise de utilizator care permit impunerea asupra documentului a oricărei vederi imaginabile.

Există două metode de parcurgere a arborelui documentului DOM Level 2: cu un `Iterator` sau cu un `TreeWalker`.

Un `Iterator` vizualizează documentul ca o listă simplă și ordonată de noduri. Deoarece arborele documentului poate fi modificat în timpul traversării listei, poziția lui `Iterator` este întotdeauna între noduri sau înainte de start, respectiv după încheiere. Aceasta înseamnă că `nextNode()` și `previousNode()` returnează întotdeauna ceva, fie nul dacă se încearcă trecerea dincolo de locațiile terminale ale arborelui, fie nodul următor. Deci parcurgerile pot găsi un nod diferit atunci când se inversează direcția de la ultimul nod vizitat. Aceasta nu este o eroare.

Un `TreeWalker` vizualizează documentul ca un arbore, deci parcurgerea arborelui poate implica atât relații părinte, frate și copil, cât și parcurgeri ordonate, elementare, ale nodurilor. Comentariile anterioare despre ce se întâmplă atunci când documentul de bază se modifică se aplică și pentru `TreeWalker`.

Controlul vederii logice a arborelui documentului

Ambele metode, `Iterator` și `TreeWalker`, întrebuințează indicatoare `whatToShow` și filtre pentru a determina ce părți ale arborelui să afișeze. Prin permiterea selectivă a apariției diferitelor părți ale arborelui utilizând indicatoare `whatToShow` și filtre, porțiunile care nu sunt esențiale pot fi omise, ca și când nu ar exista. Aceasta simplifică de exemplu căutarea de text, deoarece documentul poate fi prezentat ca un fișier text secvențial plat prin dezactivarea tuturor indicatoarelor, cu excepția indicatorului `SHOW_TEXT`. De asemenea, dacă din anumite motive trebuie să vizualizați toate instrucțiunile de prelucrare, dezactivați toate indicatoarele, mai puțin `SHOW_PROCESSING_INSTRUCTION`. Dacă nevoile dumneavoastră specifice de prelucrare necesită un filtru care nu a fost încă furnizat de un indicator `whatToShow`, atunci trebuie să scrieți un filtru.

Indicatoare `whatToShow`

Un indicator `whatToShow` constă în zero sau mai multe indicatoare alese din lista următoare, preluată din candidatul de Recomandare DOM Level 2. Versiunea completă poate fi studiată la `http://www.w3.org/tr/`. Lista următoare descrie ce fac diversele indicatoare `whatToShow`:

- `SHOW_ALL` – Arată toate nodurile. Această valoare spune unui `Iterator` sau `TreeWalker` să parcurgă toate nodurile documentului, mai puțin atributele, entitățile și notațiile care nu sunt văzute, deoarece nu fac parte din arborele documentului.
- `SHOW_ELEMENT` – Arată nodurile element. Această valoare spune unui `Iterator` sau `TreeWalker` să parcurgă toate nodurile element ale arborelui sărind peste instrucțiunile de prelucrare și peste alte tipuri de noduri, ca și când nu ar exista.
- `SHOW_ATTRIBUTE` – Arată nodurile atribut. Această valoare este semnificativă numai atunci când se creează un `Iterator` având ca rădăcină un nod atribut. A avea ca rădăcină un nod atribut implică faptul că nodul atribut apare pe prima poziție în lista de iterații. Deoarece atributele nu sunt de fapt părți ale arborelui documentului, în alte situații nu vor apărea atunci când se iterează în arborele documentului.
- `SHOW_TEXT` – Arată nodurile text. Această valoare spune unui `Iterator` sau `TreeWalker` să traverseze nodurile text ale arborelui sărind peste elemente și alte noduri ca și când nu ar exista, dacă acestea nu sunt selectate de un alt indicator.
- `SHOW_CDATA_SECTION` – Arată nodurile `CDATASection`. Această valoare spune unui `Iterator` sau `TreeWalker` să parcurgă nodurile `CDATA` ale arborelui sărind peste elemente și alte noduri ca și când nu ar exista, dacă acestea nu sunt selectate de un alt indicator.
- `SHOW_ENTITY_REFERENCE` – Arată nodurile `EntityReference`. Această valoare spune unui

Iterator sau TreeWalker să parcurgă nodurile entitate referință ale arborelui sărind peste elemente și alte noduri ca și când nu ar exista, dacă acestea nu sunt selectate de un alt indicator.

- SHOW_ENTITY – Arată nodurile Entity. Această valoare este semnificativă numai atunci când se creează un Iterator având ca rădăcină un nod Entity individual. A avea ca rădăcină un nod Entity presupune ca nodul Entity să apară pe prima poziție în lista de iterații. Deoarece entitățile nu sunt de fapt părți ale arborelui documentului, în celelalte situații nu vor apărea atunci când se iterează în arborele documentului.
- SHOW_PROCESSING_INSTRUCTION – Arată nodurile ProcessingInstruction. Această valoare spune unui Iterator sau TreeWalker să parcurgă nodurile instrucțiuni de prelucrare ale arborelui sărind peste elemente și alte noduri ca și când nu ar exista, dacă acestea nu sunt selectate de un alt indicator.
- SHOW_COMMENT – Arată nodurile Comment. Această valoare spune unui Iterator sau TreeWalker să parcurgă nodurile comentariu ale arborelui sărind peste elemente și alte noduri ca și când nu ar exista, dacă acestea nu sunt selectate de un alt indicator.
- SHOW_DOCUMENT – Arată nodurile Document. Această valoare spune unui Iterator sau TreeWalker să parcurgă nodurile document ale arborelui sărind peste elemente și alte noduri ca și când nu ar exista, dacă acestea nu sunt selectate de un alt indicator.
- SHOW_DOCUMENT_TYPE – Arată nodurile DocumentType. Această valoare spune unui Iterator sau TreeWalker să parcurgă nodurile tip de document ale arborelui sărind peste elemente și alte noduri ca și când nu ar exista, dacă acestea nu sunt selectate de un alt indicator.
- SHOW_DOCUMENT_FRAGMENT – Arată nodurile DocumentFragment. Această valoare spune unui Iterator sau TreeWalker să parcurgă nodurile fragment de document ale arborelui sărind peste elemente și alte noduri ca și când nu ar exista, dacă acestea nu sunt selectate de un alt indicator.
- SHOW_NOTATION – Arată nodurile Notation. Această valoare este semnificativă numai atunci când se creează un Iterator având ca rădăcină un nod Notation individual. A avea ca rădăcină un nod Notation implică faptul că nodul Notation apare pe prima poziție în lista de iterații. Deoarece notațiile sunt referințe externe și nu sunt de fapt părți ale arborelui documentului, în celelalte situații nu vor apărea atunci când se iterează în arborele documentului.

Chiar dacă un nod nu trece de testul whatToShow, copiii săi sunt totuși evaluați atât de Iterator, cât și de TreeWalker, pentru a fi incluși în vedere.

Filtre

Filtrele sunt activate printr-un cârlig simplu care apelează o funcție definită de utilizator. În DOM Level 2 nu sunt încorporate alte filtre în afară de indicatoarele whatToShow, care utilizează un mecanism complet diferit pentru a modifica vederea logică a documentului. Există o singură metodă asociată unui filtru, acceptNode(), care returnează una dintre următoarele trei valori:

- FILTER_ACCEPT – Această valoare îi comunică lui Iterator sau lui TreeWalker să accepte nodul. Aceasta înseamnă că nodul apare în vederea logică a arborelui. Dacă această valoare nu este returnată, atunci nodul nu apare, iar comportamentul lui TreeWalker este determinat prin returnarea uneia dintre celelalte două valori rămase.

- FILTER_SKIP – Această valoare îi comunică lui Iterator sau lui TreeWalker să sară peste nod, dar să prelucreze descendenții săi dacă se utilizează un TreeWalker. Iteratorii prelucrează întotdeauna următorul nod vizibil, deci FILTER_SKIP și FILTER_REJECT sunt echivalente pentru iteratori. Acest lucru are sens dacă vă gândiți puțin, deoarece iteratorii nu au, în vederea lor unidimensională, conceptul de noduri copii.
- FILTER_REJECT – Această valoare îi comunică lui Iterator sau lui TreeWalker să elimine nodul și să ignore copiii săi dacă se folosește un TreeWalker. Iteratorii prelucrează întotdeauna următorul nod vizibil, deci FILTER_SKIP și FILTER_REJECT sunt echivalente pentru iteratori. Acest lucru are sens dacă vă gândiți puțin, deoarece iteratorii nu au, în vederea lor unidimensională, conceptul de noduri copii.

În general, FILTER_SKIP este mai sigur în utilizare atunci când un nod trebuie ignorat, deoarece permite atât lui Iterator cât și lui TreeWalker să utilizeze același filtru și să se comporte într-o manieră asemănătoare. În situații speciale, poate veți dori să utilizați FILTER_REJECT, dar trebuie să fiți conștient că filtrele dau rezultate diferite atunci când sunt utilizate atât pentru Iterator, cât și pentru TreeWalker. Dar pentru situațiile în care este necesară eliminarea unei ramuri întregi a unui arbore în funcție de starea unui anumit nod, FILTER_REJECT este cea mai bună alegere.

Următorul fragment Java instanțiază un NodeFilter() care sare peste elementele Image din orice motiv pe care vi-l puteți imagina:

```
class ImageFilter implements NodeFilter {
  short acceptNode(Node nodevalue) {
    if (nodeValue instanceof Element) {
      Element elementvalue = (element)nodevalue;
    if (! elemantvalue.getNodeName().equals("Image"))
      return FILTER_SKIP;
    if (elementvalue.getAttributeNode("NAME") != null) {
      return FILTER_ACCEPT;
      }
    }
  return FILTER_SKIP;
  }
}
```

Notă Această secțiune (și altele) se bazează pe candidatul de Recomandare DOM Level 2 care, în calea sa spre statutul de Recomandare, poate suferi unele modificări. Ca întotdeauna, documentul curent, așa cum este prezentat la http://www.w3.org/TR/, este cel mai demn de încredere.

Filtrele sunt o metodă extrem de puternică pentru transformarea arborelui documentului. Cu un document proiectat corect, un filtru adecvat poate modifica documentul astfel încât acesta să corespundă aproape oricărui dispozitiv de afișare sau oricărei nevoi de interactivitate cu alte documente. De exemplu, un plan de lecție poate conține teste opționale cu răspunsuri care ar fi afișate în funcție de un criteriu extern, lecturi suplimentare pentru cei interesați să aprofundeze tema lecției sau chiar limbi alternative pentru clasele bilingve, toate într-un singur document.

Utilizarea DOM la nivel de nod

DOM întreține două vederi ale unui document XML. Prima, o vedere plată în care orice este un nod și poate fi manipulat în termenii poziției în arborele documentului. Aceste metode de accesare a obiectului document sunt asociate cu interfața `Node`, iar în DOM Level 2 cu interfețele `Traversal`. Cea de-a doua utilizează metodele asociate cu toate celelalte interfețe disponibile și este prezentată în secțiunea următoare.

Sugestie de la LeeAnne.com Words to weave by

DOM la nivel de nod este mai simplu în utilizarea sa de cicluri de prelucrare decât vederea documentului complet orientată spre obiecte din DOM Level 2 sau superior, deci poate fi mai adecvat pentru medii în care performanța este esențială.

Multe interfețe la nivel de nod sau complet orientate spre obiecte se reproduc parțial una pe cealaltă, dar ambele versiuni sunt păstrate pentru a menține nivelul general de satisfacție al diferiților clienți ai DOM.

Atribute nod

Atributele nod permit evaluarea nodului curent sau context și a anumitor noduri înrudite cu acesta. Interfețele nod conțin următoarele atribute:

- `nodeName` – Returnează numele nodului context.
- `nodeValue` – Returnează valoarea sau conținutul nodului curent sau context.
- `nodeType` – Returnează un cod numeric indicând tipul nodului curent sau context.
- `parentNode` – Returnează cel mai apropiat părinte vizibil al nodului context, sau nul dacă nodul curent nu are un părinte. Vizibilitatea este afectată de indicatoarele `whatToShow`.
- `childNodes` – Returnează un `NodeList` conținând numele tuturor nodurilor copil ale nodului curent sau context, sau nul dacă nodul curent nu are nici un copil.
- `firstChild` – Returnează primul copil al nodului context, sau nul dacă nodul curent nu are nici un copil.
- `lastChild` – Returnează ultimul copil al nodului context, sau nul dacă nodul curent nu are nici un copil.
- `previousSibling` – Returnează fratele precedent al nodului context, sau nul dacă nu există un astfel de nod.
- `nextSibling` – Returnează următorul frate al nodului context, sau nul dacă nu există un astfel de nod.
- `attributes` – Returnează un `NamedNodeMap` conținând o listă a atributelor aparținând nodului curent sau context.
- `ownerDocument` – Returnează obiectul document care conține nodul context, sau nul dacă nodul curent este un document.
- *`namespaceURI`* – Returnează URI-ul zonei de nume asociate nodului context, sau nul dacă acesta nu există.
- *`prefix`* – Returnează prefixul zonei de nume asociate nodului curent sau context, sau nul dacă acesta nu există.
- *`localName`* – Returnează partea locală a numelui nodului curent sau context, adică numele fără nici un prefix de zonă de nume. Dacă nu există nici un prefix de zonă de nume, acesta este echivalentul exact al lui `nodeName`.

Interfețele nod au următoarele metode, utilizate pentru a manipula modelul document obiect:

- `insertBefore(newChild, namedChild)` – Inserează un copil înaintea unui nod `namedChild`.
- `replaceChild(newChild, namedChild)` – Înlocuiește un nod `namedChild` cu altul.
- `removeChild(namedChild)` – Elimină un nod `namedChild`.
- `appendChild(newChild)` – Adaugă un nou nod copil la lista copiilor.
- `hasChildNodes()` – Un test booleean pentru a determina dacă nodul curent sau context are sau nu copii.
- `cloneNode(deep)` – Copiază nodul curent sau context incluzând opțional, în mod recursiv, nodurile copil, dacă parametrul `deep` este setat.
- *`supports(featureName)`* – Un test booleean pentru a determina dacă o implementare DOM suportă o anumită caracteristică.

Elementele scrise *cursiv* sunt definite în DOM Level 2.

La lista anterioară ar trebui adăugate atributul și metoda următoare din interfața `NodeList`:

- `length` – Returnează lungimea lui `NodeList` sau nul.

Mai există o metodă suplimentară la interfața `NodeList`

- `item(indexNumber)` – Returnează un articol indexat dintr-un `NodeList`

și atributul și metodele următoare din interfața `NamedNodeMap`:

- `length` – Returnează lungimea lui `NamedNodeList` sau nul

și metodele:

- `getNamedItem(name)` – Returnează un articol cu nume din harta de noduri.
- `setNamedItem(nodeArgument)` – Adaugă la hartă un articol cu nume.
- `removeNamedItem(name)` – Elimină din hartă un articol cu nume.
- `item(index)` – Returnează un articol indexat din hartă.
- *`getNamedItemNS(nameSpaceURI, localName)`* – Returnează din hartă un articol cu nume calificat pentru zonă de nume.
- *`setNamedItemNS12(nameSpaceURI, localName)`* – Adaugă la hartă un articol cu nume calificat pentru zonă de nume.
- *`removeNamedItemNS(nameSpaceURI, localName)`* – Elimină din hartă un articol cu nume calificat pentru zonă de nume.

Elementele scrise *cursiv* sunt definite în DOM Level 2.

DOM Level 2 adaugă la acest set de caracteristici de bază următoarele interfețe de parcurgere:

- `NodeIterator` – Are atributele `whatToShow`, `filter` și `expandEntityReferences`. Are metodele `nextNode` și `previousNode`.
- `TreeWalker` – Are atributele `whatToShow`, `filter`, `expandEntityReferences` și `currentNode`. Are metodele `parentNode`, `firstChild`, `lastChild`, `previousSibling`, `nextSibling`, `previousNode` și `nextNode`.
- `NodeFilter` – Are metoda `acceptNode`.
- `DocumentTraversal` – Are metodele `createNodeIterator` și `createTreeWalker`.

Interfețele de parcurgere prezentate aici sunt definite mai detaliat în secțiunea precedentă, „Un copac poate ascunde o pădure"; dar pentru un caracter complet sunt trecute și aici.

Utilizarea unui DOM orientat spre obiecte

În afară de interfețele nod, fiecare interfață DOM este complet orientată spre obiecte. Deoarece multe dintre aceste metode sunt destul de scumpe, necesitând conversii forțate sau alte extravaganțe de prelucrare, interfețele la nivel de nod permit un acces relativ ieftin la document, fără cheltuielile suplimentare necesare pentru deplina orientare spre obiecte.

Majoritatea claselor DOM utilizate de obicei sunt moștenite din clasa nod. Deci elementele, atributele, notațiile, entitățile și chiar documentul în sine sunt noduri. Aceasta este într-adevăr o afirmație banală, dar edificatoare atunci când vă amintiți că un document XML are o structură de arbore și, din acest motiv, poate fi reprezentat ca un arbore. Deoarece orice apare în arbore este fie un nod, fie un arc care cuprinde două noduri, rezultă că orice lucru tangibil dintr-un document XML este un nod în acel arbore. Aceasta permite ca întregul document să fie accesat și manipulat utilizând metode DOM.

Excepțiile de la această regulă generală sunt clasele care sunt generate și utilizate intern de DOM. Deci `NodeList` și `NamedNodeMap` nu sunt de fapt noduri, deoarece reprezintă metainformații despre noduri pe care un DOM le utilizează pentru prelucrări și nu noduri în sine. De asemenea, `DOMException` și `DOMImplementation` sunt metainformații referitoare la starea procesului DOM în sine, și nu sunt chiar noduri. Atunci când procesul DOM se încheie, aceste abstracții dispar fără urmă lăsând nodurile propriu-zise întrupate în diversele elemente, atribute și diverse alte părți care formează un document XML.

Interfețe fundamentale

Așa-numitele interfețe fundamentale sunt acelea care sunt necesare pentru manevrarea HTML. În ciuda faptului că sunt necesare și pentru XML sau orice alt document structurat, XML necesită aptitudini extinse, pe care le găsiți, firește, în secțiunea următoare, „Interfețe extinse".

HTML poate fi prelucrat în mod diferit decât un document XML general, deoarece DTD-ul este predefinit și poate fi implementat în mod direct în agentul utilizator. De fapt, aceasta este cea mai uzuală metodă de implementare a unui agent utilizator HTML. Deoarece acest subset al funcționalității DOM adresează același domeniu de probleme ca HTML, interfețele fundamentale sau principale nu sunt tratate în detaliu.

Sugestie de la LeeAnne.com Words to weave by

Deși interfețele și metodele DOM sunt definite într-o manieră orientată spre obiecte, o interfață nu este neapărat o clasă. Interfața este necesară numai pentru a expune metode care corespund celor numite, dar nu necesită implementarea unei clase corespunzătoare. Aceasta permite, dacă este nevoie, implementări reduse ale DOM peste aplicații moștenite.

Iată câteva interfețe fundamentale:

- `DOMException(code)` – Furnizează o interfață standard la o listă de mesaje de eroare pe care le poate da un modul utilizator. Unele limbaje nu suportă excepțiile, iar aceeași listă de erori poate fi prelucrată utilizând orice mecanism standard de erori care există în acel limbaj.
- `ExceptionCode` – O listă de constante întregi utilizată de `DOMException`.
- `DOMImplementation()` – Furnizează o metodă care permite unei aplicații utilizator să testeze aptitudinile implementării DOM. În DOM Level 2, aceasta furnizează de asemenea metode independente de implementare pentru crearea de documente XML.
- `DocumentFragment` – Furnizează un suport de categorie ușoară pentru părți relativ mici ale unui document, ceea ce simplifică mutarea de părți ale documentului sau crearea de fragmente de document care să fie inserate în arborele documentului după terminarea prelucrării.
- `Document` – Furnizează metode pentru adresarea întregului document și crearea de elemente, noduri text, comentarii, instrucțiuni de prelucrare și altele care nu pot exista în afara contextului unui document.
- `Node` – Furnizează metode pentru manipularea nodurilor.
- `NodeList` – Furnizează un suport pentru o listă de noduri și o metodă pentru returnarea unui singur articol din acea listă.
- `NamedNodeMap` – Furnizează metode de utilizare și întreținere a listelor de noduri cu nume.
- `CharacterData` – Furnizează metode care extind `Node`.
- `Attr` – Furnizează o abstractizare a unui atribut.
- `Element` – Furnizează abstractizarea unui element și câteva metode pentru manipularea atributelor.
- `Text` – Furnizează o metodă de separare a datelor text în părți mai mici.
- `Comment` – Furnizează o abstractizare a unui comentariu.

DOM Level 2 adaugă câteva metode la registrul de module principale de interfață, dar în rest, au rămas neschimbate.

Interfețe extinse

Interfețele extinse sunt acelea care sunt necesare pentru manevrarea documentelor XML și asemănătoare cu XML, și care nu sunt întâlnite niciodată la prelucrarea documentelor HTML. Toate aceste interfețe sunt noi, deoarece sunt utilizate pentru a accesa XML și alte limbaje pentru documente. Și așa cum interfețele principale nu trebuie să fie implementate utilizând clase efective, interfețele extinse pot expune, dacă așa se dorește, numai metodele definite. Iată câteva interfețe extinse:

- `CDATASection` – Furnizează o abstractizare a unei secțiuni CDATA.
- `DocumentType` – Furnizează o abstractizare a unui tip de document.
- `Notation` – Furnizează o abstractizare a unei notații.
- `Entity` – Furnizează o abstractizare a unei entități.
- `EntityReference` – Furnizează o abstractizare a unei entități referință.
- `ProcessingInstruction` – Furnizează o abstractizare a unei instrucțiuni de prelucrare.

DOM Level 2 adaugă câteva atribute noi la vreo două module de interfață, dar interfețele extinse sunt, altfel, neschimbate. Recomand din toată inima citirea specificațiilor, deși unele părți ale lor sunt oarecum obscure. Ele au fragmente de cod și asocieri complete Java și ECMAScript pentru edificarea programatorilor și sunt întotdeauna cea mai autorizată sursă de informații.

Utilizarea explicațiilor din această carte, combinate cu descrierile succinte din specificațiile W3C, ar trebui să vă permită un compromis bun între utilitate și vorbăria excesivă a standardelor W3C.

Metode DOM

Deși o discuție completă referitoare la orientarea spre obiecte nu face subiectul acestei cărți, ar trebui să știți că metodele DOM (ca orice metodă orientată spre obiecte) sunt căi foarte controlate pentru accesarea sau modificarea informației conținute într-un obiect.

Orientarea spre obiecte își are începuturile în simularea proceselor, deci nu ar trebui să vă surprindă faptul că obiectele structurii documentului, accesate prin metode DOM, sunt dinamice. Deci, dacă un nod conținut într-un `NodeList` existent este șters sau dacă se adaugă un nod nou, atunci `NodeList` reflectă acel lucru fără a fi nevoie de nici un efort din partea dumneavoastră. De asemenea, adăugarea sau ștergerea de noduri în sau din arborele documentului sunt operații transparente pentru metodele de navigare și de parcurgere.

Acest lucru este uneori greu de înțeles pentru programatorii în limbaje procedurale, deoarece structurile lor de date sunt statice, necesitând actualizări explicite pentru a rămâne la zi.

O altă caracteristică orientată spre obiecte este moștenirea, deci o clasă nod (o clasă este mai mult sau mai puțin un obiect, dar văzut ca un șablon pentru crearea altor obiecte) are asociate anumite metode. Deoarece elementele, de exemplu, sunt instanțieri ale unei clase nod, ele moștenesc toate metodele și atributele nodului și trebuie adăugate numai câteva metode specifice nodurilor element.

Totuși, acest lucru poate fi și o sursă de erori. Un nod text nu poate fi un părinte pentru nici un nod copil, chiar dacă metodele nodului care se referă la copii există în nodul text, deoarece el moștenește aceste metode de la clasa lui părinte. Încercarea de a crea un copil pentru un nod text lansează o excepție. Atunci când citiți specificațiile W3C, aceste lucruri nu vor fi scoase în evidență în descrierea fiecărei interfețe, dar trebuie să știți că majoritatea nodurilor documentului sunt moștenite de undeva. Din păcate, în recomandarea DOM, informațiile despre moștenirea nodurilor sunt dispersate în diverse părți. Deși descrierile din recomandare ale interfeței formale vă spun dacă un obiect este sau nu moștenit, pentru o referință comodă urmează o serie de subsecțiuni conținând liste simple, care arată, dacă este cazul, de unde este moștenit un obiect:

Moștenirea DOM principală

Moștenirea este un termen orientat spre obiecte, care se referă la însușirea unui obiect de a moșteni atribute și metode de la clasa strămoș din care este derivat. Deoarece DOM nu are nevoie de implementarea efectivă a majorității interfețelor utilizând clasele, s-ar putea să fie nevoie ca moștenirea să fie simulată prin alte mijloace. Aceste obiecte fac parte din modulul DOM principal:

- `DocumentFragment` este moștenit de la `Node`
- `Document` este moștenit de la `Node`
- `CharacterData` este moștenit de la `Node`
- `Attr` este moștenit de la `Node`
- `Element` este moștenit de la `Node`
- `Text` este moștenit de la `CharacterData`
- `Comment` este moștenit de la `CharacterData`
- `CDATASection` este moștenit de la `Text`
- `DocumentType` este moștenit de la `Node`

- `Notation` este moștenit de la `Node`
- `Entity` este moștenit de la `Node`
- `EntityReference` este moștenit de la `Node`
- `ProcessingInstruction` este moștenit de la `Node`

Moștenirea DOM CSS

Aceste obiecte fac parte din modulul DOM CSS:

- `CSSStyleSheet` este moștenit de la `stylesheets::style sheet`
- `CSSStyleRule` este moștenit de la `CSSRule`
- `CSSMediaRule` este moștenit de la `CSSRule`
- `CSSFontFaceRule` este moștenit de la `CSSRule`
- `CSSPageRule` este moștenit de la `CSSRule`
- `CSSImportRule` este moștenit de la `CSSRule`
- `CSSCharsetRule` este moștenit de la `CSSRule`
- `CSSUnknownRule` este moștenit de la `CSSRule`
- `CSSPrimitiveValue` este moștenit de la `CSSValue`
- `CSSValueList` este moștenit de la `CSSValue`
- `ViewCSS` este moștenit de la `views::AbstractView`
- `DocumentCSS` este moștenit de la `stylesheets::DocumentStyle`
- `DOMImplementationCSS` este moștenit de la `DOMImplementation`
- `CSS2Azimuth` este moștenit de la `CSSValue`
- `CSS2BackgroundPosition` este moștenit de la `CSSValue`
- `CSS2BorderSpacing` este moștenit de la `CSSValue`
- `CSS2CounterReset` este moștenit de la `CSSValue`
- `CSS2CounterIncrement` este moștenit de la `CSSValue`
- `CSS2Cursor` este moștenit de la `CSSValue`
- `CSS2PlayDuring` este moștenit de la `CSSValue`
- `CSS2PageSize` este moștenit de la `CSSValue`
- `HTMLElementCSS` este moștenit de la `html::HTMLElement`

Moștenirea DOM Event

Aceste obiecte fac parte din modulul de evenimente DOM:

- `UIEvent` este moștenit de la `Event`
- `MouseEvent` este moștenit de la `UIEvent`
- `KeyEvent` este moștenit de la `UIEvent`
- `MutationEvent` este moștenit de la `Event`

Moștenirea DOM HTML

Aceste obiecte fac parte din modulul DOM HTML:

- `HTMLDOMImplementation` este moștenit de la `DOMImplementation`
- `HTMLDocument` este moștenit de la `Document`
- `HTMLElement` este moștenit de la `Element`
- `HTMLHtmlElement` este moștenit de la `HTMLElement`

- HTMLHeadElement este moștenit de la HTMLElement
- HTMLLinkElement este moștenit de la HTMLElement
- HTMLTitleElement este moștenit de la HTMLElement
- HTMLMetaElement este moștenit de la HTMLElement
- HTMLBaseElement este moștenit de la HTMLElement
- HTMLIsIndexElement este moștenit de la HTMLElement
- HTMLStyleElement este moștenit de la HTMLElement
- HTMLBodyElement este moștenit de la HTMLElement
- HTMLFormElement este moștenit de la HTMLElement
- HTMLSelectElement este moștenit de la HTMLElement
- HTMLOptGroupElement este moștenit de la HTMLElement
- HTMLOptionElement este moștenit de la HTMLElement
- HTMLInputElement este moștenit de la HTMLElement
- HTMLTextAreaElement este moștenit de la HTMLElement
- HTMLButtonElement este moștenit de la HTMLElement
- HTMLLabelElement este moștenit de la HTMLElement
- HTMLFieldSetElement este moștenit de la HTMLElement
- HTMLLegendElement este moștenit de la HTMLElement
- HTMLULListElement este moștenit de la HTMLElement
- HTMLOListElement este moștenit de la HTMLElement
- HTMLDListElement este moștenit de la HTMLElement
- HTMLDirectoryElement este moștenit de la HTMLElement
- HTMLMenuElement este moștenit de la HTMLElement
- HTMLLIElement este moștenit de la HTMLElement
- HTMLDivElement este moștenit de la HTMLElement
- HTMLParagraphElement este moștenit de la HTMLElement
- HTMLHeadingElement este moștenit de la HTMLElement
- HTMLQuoteElement este moștenit de la HTMLElement
- HTMLPreElement este moștenit de la HTMLElement
- HTMLBRElement este moștenit de la HTMLElement
- HTMLBaseFontElement este moștenit de la HTMLElement
- HTMLFontElement este moștenit de la HTMLElement
- HTMLHRElement este moștenit de la HTMLElement
- HTMLModElement este moștenit de la HTMLElement
- HTMLAnchorElement este moștenit de la HTMLElement
- HTMLImageElement este moștenit de la HTMLElement
- HTMLObjectElement este moștenit de la HTMLElement
- HTMLParamElement este moștenit de la HTMLElement
- HTMLAppletElement este moștenit de la HTMLElement
- HTMLMapElement este moștenit de la HTMLElement
- HTMLAreaElement este moștenit de la HTMLElement
- HTMLScriptElement este moștenit de la HTMLElement

- HTMLTableElement este moștenit de la HTMLElement
- HTMLTableCaptionElement este moștenit de la HTMLElement
- HTMLTableColElement este moștenit de la HTMLElement
- HTMLTableSectionElement este moștenit de la HTMLElement
- HTMLTableRowElement este moștenit de la HTMLElement
- HTMLTableCellElement este moștenit de la HTMLElement
- HTMLFrameSetElement este moștenit de la HTMLElement
- HTMLFrameElement este moștenit de la HTMLElement
- HTMLIFrameElement este moștenit de la HTMLElement

Structura DOM

DOM nu are o structură efectivă, deși este bazat pe o vedere a documentului XML orientată spre obiecte și cu o structură de arbore. Altfel spus, valorile returnate de interfețe ar fi true (adevărat) dacă s-ar referi la un document care ar fi de fapt un arbore. Dar DOM nu presupune efectiv că un document *este* un arbore, ci insistă să îl trateze ca și cum ar fi unul.

Aceasta nu este o deosebire nefondată. Multe aplicații ale XML materializează fără îndoială o bază de date subiacentă sau o structură de informații care este posibil să nu aibă nimic în comun cu nici un fel de arbori.

DOM permite unui dezvoltator XML să își acopere structura de bază, oricare ar fi aceasta, cu un înveliș subțire de metode care transformă acea structură într-un arbore pentru utilizarea publică, chiar dacă structura de bază și mecanismele rămân nemodificate.

Aceasta este baza utilității XML ca mediu de transfer, permițând utilizarea fluxurilor de date XML pentru a transfera informații la și de la sisteme de baze de date diferite. Este, de asemenea, baza valorii XML ca format de afișare, deoarece mai multe baze de date pot fi afișate și manipulate cu un singur mecanism.

Studiul capriciilor din DOM

Există multe caracteristici ale DOM care se copiază reciproc sau au fost adăugate la insistența unui anumit furnizor, reflectând concepția acestuia despre utilizarea XML. Microsoft, de exemplu, a cerut și a obținut un atribut ID pentru suportul insulelor de date (Data Island) – un mod de a îngloba fragmente XML într-un document HTML utilizând încă o etichetă HTML, brevetată Microsoft, <XML>.

Analizoarele XML de la Microsoft au un mod nestandard de încărcare a unui document XML, datorită faptului că specificația DOM originală nu se referea la această problemă. Dar celelalte analizoare XML au ținut pasul cu standardul aflat în dezvoltare, în timp ce Microsoft a hotărât să rămână cu implementarea sa mai veche până când DOM Level 2 va deveni o Recomandare; acest lucru creează o problemă atunci când se scriu programe pentru ambele standarde.

Microsoft a creat o metodă standard pentru continuarea arborelui XML în documentul XML, lucru evitat cu premeditare chiar și în schița de lucru DOM Level 2 a W3C. Este probabil o idee bună să nu se depoziteze în cache structura documentului în acest mod,

dar este tentant, datorită sporului de performanță. Un mecanism mai sofisticat de depozitare în cache ar realiza acest lucru fără intervenția programatorului.

Microsoft a creat o metodă de interceptare a erorilor de analizare obișnuite. Și lista poate continua, dar dumneavoastră ați înțeles deja ideea. Din păcate, utilizarea extensiilor Microsoft în documentele dumneavoastră va face ca acest cod să se blocheze atunci când este utilizat pe orice altă platformă, un comportament distructiv care, pe termen lung, nu prea găsește adepți.

Pentru a evita munca obositoare de a testa browsere și de a schimba codul ca să se potrivească mediului, trebuie să codați un subset al aptitudinilor DOM suportate identic pe mai multe platforme. S-a estimat că pentru tratarea diferențelor dintre browsere este nevoie de 25% sau mai mult din bugetul proiectării Web, deci puteți vedea cu ușurință impulsul dat de costuri pentru trecerea cât mai rapidă la standarde comune.

Tratarea excepţiilor

Fiind orientat spre obiecte, este evident că DOM lansează excepții. Excepțiile trebuie interceptate în codul utilizatorului, deci este de datoria dumneavoastră să evitați, dacă este posibil, trucurile de codare care pot genera excepții. În particular, filtrele nu sunt împiedicate să schimbe arborele documentului, deși în majoritatea cazurilor nu este recomandat deoarece reîntoarcerea de la un filtru poate găsi documentul într-o stare nedefinită sau ambiguă. Modul exact de tratare a excepțiilor nu face subiectul acestei cărți. Trebuie să tratați excepțiile în conformitate cu convențiile utilizate de limbajul folosit pentru accesarea DOM-ului. În Java, sintaxa generală este:

```
try {
  myRoutine();
  }
  catch (myException exceptionVariable) {
    System.out.println("Error Message:" + exceptionVariable.data);
    }
```

Experiența proprie poate diferi în funcție de limbajul utilizat și de ceea ce doriți să faceți cu tipul de excepție descoperit de handlerul dumneavoastră. Este posibil să recuperați sau să ignorați multe excepții, sau să încercați strategii alternative până când ceea ce v-ați propus va funcționa. Deoarece tratarea excepțiilor este în general mai ieftină din punctul de vedere al timpului de prelucrare decât încercarea de a scrie un cod de pază pentru fiecare rutină, pentru a trata orice sursă de erori posibilă, dintre care multe s-ar putea să nu apară niciodată în lucrul efectiv, un handler de excepții vă oferă un plan de siguranță pentru situațiile când lucrurile nu merg cum trebuie. De asemenea, utilizarea handlerelor de excepții vă permite să centralizați rutine de erori obișnuite, permițând aplicației dumneavoastră să rămână suplă și elegantă și în același timp să se restabilească după erori neobișnuite.

Studii de caz

Deoarece DOM este o interfață de programare, teoria spune că un program proiectat să opereze pe un document XML pe o platformă va opera exact în același mod pe orice platformă. Aceasta ridică problema suportului limbajului de programare, deoarece este

un lucru obișnuit pentru un anumit limbaj să fie implementat în moduri puțin diferite pe platforme diferite, iar noi am văzut deja că în momentul de față DOM este fragmentat într-o oarecare măsură la granița Microsoft/Toți ceilalți.

Dar această ultimă problemă ar trebui să se rezolve măcar parțial dacă cei de la Microsoft ar implementa suportul promis pentru DOM-ul definit de W3C, și nu pentru DOM-ul propriu. Totuși, dată fiind existența numărului mare de pagini și instrumente proiectate să accepte mai vechile lor produse care nu sunt standard, acest lucru poate fi realizat cu greu fără a bloca un număr mare de pagini moștenite și fără a îndepărta utilizatori care, în trecut, au depins de ele. Ei se află între două focuri și, din acest motiv, va fi interesant de văzut cum se vor (ne vom) descurca.

Probabil că, în termeni practici, va trebui să luăm în considerare această istorie a incompatibilității pentru viitorul apropiat, în afară de cazul în care ne vom limita navigarea la siturile cele mai noi și actualizate cu regularitate.

Iar dacă este neapărat necesară schimbarea limbajelor de programare pentru a suporta platforme multiple și pentru a lua în considerare multiplele versiuni ale „standardului", atunci se pierde mult din valoarea DOM. Deci DOM strigă după un limbaj de programare cu adevărat neutru, care să completeze propriul design independent de platformă, iar istoria limbajului Java este una plină de învățăminte. Anunțat odată ca fiind universal, cel puțin un furnizor menționat în această secțiune l-a „îmbunătățit" până în punctul în care codul Java scris pentru platforma lor nu a mai funcționat pe nici o altă platformă. La rândul său, adevăratul Java nu ar funcționa pe platforma lor fără o restricționare severă a lucrurilor pe care le face.

XML promite să reușească acolo unde alte încercări de standardizare au eșuat, deoarece greutatea istoriei este în spatele unei eventuale standardizări atunci când tehnologia devine suficient de importantă. Unele părți ale XML au fost deja mutate pe siliciu, iar cipurile acceleratoare XML promit să facă pentru schimbul de documente XML ceea ce au făcut acceleratoarele grafice pentru jocurile pe calculator. Deci putem privi cu optimism spre viitor, în ciuda unei istorii lungi de dezamăgiri amare.

SAX

(The Simple API for XML) – API simplu pentru XML

CAPITOLUL **11**

În acest capitol

Utilizarea SAX în locul DOM
Analizoare SAX
SAX în detaliu
Tehnici orientate spre obiecte
Depanarea SAX

Utilizarea SAX în locul DOM

Dată fiind complexitatea DOM, era nevoie de un fel de instrument pentru a permite oamenilor obișnuiți să prelucreze fișiere XML fără a construi un arbore riguros al documentului și fără a da din mâini în jurul lui ca o maimuță extraordinar de talentată, așa cum face DOM. Se pare că există alte modalități de a privi un fișier XML. Deși maimuța DOM poate avea o privire generală ireproșabilă asupra întregii structuri, ea are nevoie de resurse bogate de memorie și de putere de prelucrare pentru a construi și a reține această informație. O furnică parcurgând metodic ramurile aceluiași arbore poate face multă muncă utilă, aproape fără nici un efort.

Exact acest lucru îl face API-ul simplu pentru XML (SAX – the Simple API for XML). Dacă parcurgeți un arbore, neacordând prea multă atenție structurii ca întreg, o puteți vizualiza ca pe o secvență de evenimente. Prima dată, găsiți rădăcina. Apoi, găsiți primul nod. Prin utilizarea unei strategii simple puteți vizita pe rând fiecare nod, acționând asupra lucrurilor găsite acolo. După ce ați vizitat fiecare nod, ați terminat.

Majoritatea programatorilor știu cum să manipuleze acest gen de operație. Este un lucru atât de obișnuit că are și un nume, un handler de evenimente. Prin parcurgerea arborelui, lucru pe care îl faceți chiar prin citirea secvențială a fișierului, datorită modului în care este structurat XML-ul, convertiți o serie complexă de relații într-un simplu flux de date. Ei bine, fluxurile de date sunt micul dejun al oricărui programator care se respectă.

Notă SAX are două variante: versiunea veche, utilizată încă de multe aplicații, și versiunea nouă, 2.0, care suportă zonele de nume și alte lucruri interesante introduse de XML. Dezvoltările noi ar trebui să utilizeze aproape întotdeauna SAX 2 deoarece zonele de nume sunt omniprezente în modelul XML, dar ar trebui să fiți capabil să vă adaptați la versiunile mai vechi ale SAX atunci când le întâlniți. Pentru cele mai noi informații din partea autorului, vizitați utilul sit Web al lui David Megginson, la `http://www.megginson.com/`.

Vizualizări arbore versus vizualizări eveniment

Atunci când construiți întregul arbore înainte de a face orice altceva, aveți avantajul că sunteți atotștiutor. Puteți răspunde la întrebări cum ar fi: „Cine este cel de-al doilea văr al acestui nod, eliminat de două ori?“ Pe de altă parte, este nevoie de ceva timp până să începeți să obțineți ceva rezultate. În primul rând, trebuie să aveți totul pus în ordine. Dacă arborele utilizează un model obiect diferit, atunci poate va trebui să o luați de la început. Și așa cum ați văzut la DOM, există suficiente goluri în specificația DOM și suficiente tentații de a face lucrurile în modul *dumneavoastră*, și nu în modul standard, astfel încât nu vă puteți baza pe găsirea aceluiași model de obiect oriunde mergeți. Deci trebuie să scrieți variante pentru viziunea asupra DOM a fiecărui browser, dacă aceasta există. Este o mare durere de cap.

Sugestie de la LeeAnne.com Words to weave by

În ciuda faptului că DOM este mult mai puternic, instrumentele conduse prin evenimente, cum este și SAX, sunt mai universale și mult mai potrivite în unele situații. Atunci când manipulați un fișier foarte mare sau când utilizați dispozitive hand-held, cheltuielile necesare pentru DOM pot fi prohibitive. Iar până când realizatorii de browsere vor cădea de acord asupra unui set comun de funcționalități, poate fi mult mai ușor să utilizați SAX, sau ceva asemănător, decât să jonglați cu numeroasele diferențe dintre implementările DOM.

SAX vă permite să vedeți cu ochii unei furnici, deci nu trebuie să vă preocupe întreaga imagine. Vedeți numai imediata vecinătate. Știți cine este părintele dumneavoastră și acest lucru este deseori suficient. Dacă aruncați o privire asupra capitolelor 12, „CSS1, CSS2, DSSSL, XSL“, și 13, „Foi de stil în cascadă și XML/XHTML“, care se ocupă de foile de stil în cascadă, veți vedea că mai multe construcții CSS se ocupă numai de această relație simplă. Puteți îmbunătăți abilitatea dumneavoastră de a manevra anumite tipuri de operații păstrând în memorie ceea ce ați aflat anterior, dar de fapt nu aveți nevoie de prea multe. Ceea ce nu puteți face (sau cel puțin nu cu ușurință) este să anticipați ceea ce urmează, deși există tehnici care vor permite o anticipare limitată fără a compromite serios simplitatea. În Figura 11.1 vedeți reprezentarea unui arbore mic în modul în care l-ar putea vedea o furnică. Părțile estompate reprezintă lucrurile pe care nu le știți. Părțile clare reprezintă ceea ce știți. Nodul umbrit reprezintă locul unde vă aflați în acest moment.

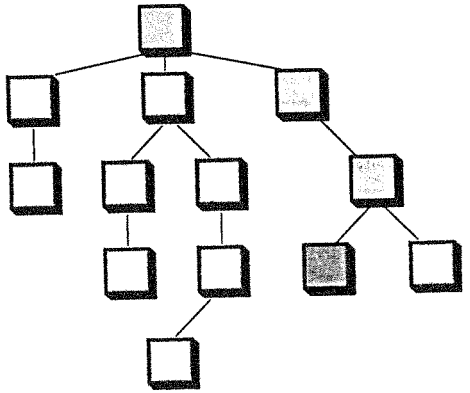


FIGURA 11.1
Un arbore așa cum este văzut de o furnică, sigură în privința liniei ancestrale și imediat înconjurătoare, dar nepăsătoare la celelalte ramuri ale arborelui familiei.

Puteți fi tentat ca în timpul parcurgerii arborelui să păstrați informații referitoare la frați, dar aceasta este o absurditate la o abordare condusă prin evenimente. Atunci când ați vizitat fiecare nod, ați adunat suficiente informații pentru a construi întregul arbore, dar le puteți utiliza numai la prelucrarea ultimei frunze a ultimei ramuri. Asemenea unei revelații pe patul de moarte, nu prea aveți ce face cu ele. Păstrarea de informații despre mai multe lucruri decât despre strămoșul direct face prelucrarea neregulată și dependentă de poziția exactă în arbore a nodului. În cazul în care cunoașteți frații anteriori, vă puteți schimba comportamentul bazându-vă pe acele informații, dar nu puteți face același lucru pentru frații următori, deoarece nu ați fost încă acolo.

Dacă aveți nevoie de informații complete pentru a realiza o anumită operație, atunci va trebui să utilizați o abordare bazată pe arbori, iar DOM, indiferent cât de slabe sunt implementările sale actuale, are cel puțin binecuvântarea și *aprobarea* W3C, deci puteți fi sigur că în final se va stabiliza la o uniformitate utilă. Cele mai importante abateri de la DOM-ul W3C există în browserul Internet Explorer de la Microsoft, care, întâmplător, este un adept important al standardului, chiar dacă nu complet, deci este destul de ușor să scrieți exclusiv pentru acea platformă. Poate nu este o idee bună, dar este aplicată de multe ori.

Versiunea următoare Netscape, planificată să încorporeze actuala rescriere open-source a funcționalității Navigator de la Mozilla.org, va implementa DOM-ul standard al W3C, spre deosebire de modelul document complet diferit, dar cu același nume, implementat în actualul browser Netscape 4.7. Din motive istorice, Netscape și-a urmat propriul drum cu o schemă de model document obiect complet diferită, în timp ce W3C lucra la standard. Deci ei au fost nevoiți să strângă din dinți și să renunțe complet la modelul lor pentru a urma DOM-ul W3C. Din fericire, nu au penetrat adânc piața, deci nu a trebuit să potolească mânia unei mase imense de utilizatori.

Notă

Vechiul model de dezvoltare a sistemelor brevetate, cu un cost inițial de proprietate relativ scăzut și cu un preț de întreținere ridicat, se pierde în favoarea sistemelor bazate pe standarde, care au atât prețul inițial cât și cel de întreținere la un nivel scăzut. Companiile pot concura la un nivel de piață mult mai mic prin menținerea unei relații continue cu clienții lor decât dacă ar încerca să îi constrângă cu soluții brevetate. Cu kit-uri de instrumente standard, cum ar fi SAX, disponibile gratuit sau la un preț mic, o companie nu mai trebuie să cheltuiască chiar atât de mult cu licențele și cu reînnoirea acestora și poate ajunge mult mai rapid pe piață. Prin acest lucru, concurența devine mult mai aprigă.

Microsoft are o problemă mult mai dificilă, deoarece este mai aproape de standard dar are milioane de utilizatori care depind de versiunea sa DOM brevetată. Migrarea Microsoft spre un model bazat pe standarde va trebui să ia în considerare baza sa de utilizatori, deci este probabil ca incompatibilitățile DOM să dăinuie și în viitorul apropiat.

Dacă tot ce doriți este să terminați un lucru cu minimum de agitație și maximum de viteză, atunci o abordare condusă prin evenimente poate fi cea mai bună, deoarece structurile grandioase de date cercetate prin DOM nici măcar nu există. O furnică poate să străbată orice fel de arbore și să facă ceva util cu el.

▶ **Vezi** „Axele XPath" la **pagina 170.**
▶ **Vezi** „Utilizarea abrevierilor XPath" la **pagina 174.**

Analizoare SAX

SAX nu este de fapt un analizor, ci o interfață de programare a aplicațiilor (API – Application Programming Interface) care furnizează un set de mecanisme de control obișnuite pentru un număr mare de analizoare, permițând programatorului să utilizeze acel analizor cu care este mai familiarizat. Vă puteți gândi la el ca la un automobil. Puteți pune sub capota preferată orice fel de motor, dobândind avantajele acelui motor fără a face modificări la caroserie sau la mecanismul de direcție. Nu este singurul procesor XML din lume condus prin evenimente. Mai există multe altele, iar acestea sunt trecute în Anexa B, „Instrumente pentru editarea și transformarea XML/XHTML", dar este suficient să studiem unul pentru a înțelege ideea generală.

▶ **Vezi** „Motoare de analizare XML" la **pagina 681.**

Sugestie de la LeeAnne.com Words to weave by

În lume există încă o mișcare puternică pentru software gratuit. Puteți găsi instrumente gratuite pentru a face aproape orice legat de programare, inclusiv sisteme de operare complete, compilatoare, depanatoare și analizoare XML.

Puteți găsi SAX în diverse forme, în funcție de limbajul cu care sunteți cel mai familiarizat. Versiunile Java și Python sunt ambele ușor de obținut și sunt *gratuite*, așa că această secțiune aruncă o privire asupra amândurora.

O implementare Python există la `http://www.stud.ifi.uio.no/~lmariusg/download/python/xml/saxlib.html`, întreținută prin amabilitatea lui Lars Marius Garshol de la Universitatea din Oslo, Norvegia. Comunitatea XML este răspândită în lumea întreagă și este foarte ușor ca atunci când începeți să lucrați în acest domeniu să vă împrieteniți cu oameni pe care probabil nu îi veți întâlni niciodată personal. SAX pentru Python necesită acces la Python, care are înglobat în versiunea de distribuție și un analizor XML. Pentru a descărca ultima versiune, încercați la situl Web Python, la `http://www.python.org`. Este disponibil pentru un număr mare de platforme și include o versiune garantată 100% de Pure Java, care ar trebui să vă permită să o rulați din orice loc în care există o portare Java.

Dacă aveți nevoie de Java SDK (Software Development Kit – kit de dezvoltare de software), vă rog să vizitați situl principal al companiei Sun, la `http://www.sun.com` și urmați legătura Java SDK. Versiunea actuală este Java 2, versiunea 1.2, care se găsește la `http://java.sun.com/products/jdk/1.2/`, dar este posibil să se modifice. SDK este gratuit.

Versiunea Java există în mai multe locuri, dar există una pe situl Microstar care oferă (contra cost) și alte instrumente XML: `http://www.microstar.com/products.html`. Defilați în jos pe pagină pentru a vedea intrările pentru motorul de analizare Ælfred și pentru SAX. Ælfred, pentru cei care nu au înțeles gluma, a fost regele saxonilor de vest din Wessex (o parte din ceea ce urma să fie Anglia) în cea de a doua jumătate a secolului IX. V-ați prins? *SAXon*?

Puteți să descărcați ambele produse, completând un formular pentru ca Microstar să știe cine sunteți, ceea ce este un preț mic pentru un produs puternic și util pe care veți fi capabil să îl utilizați și să câștigați bani cu el.

Sugestie de la LeeAnne.com Words to weave by

Multe dintre aceste situri conţin exemple software vaste, pe care le puteţi studia gratuit. Există puţine metode mai bune de a învăţa trucuri de codare în orice limbaj decât studierea unui cod existent, creat de experţi.

De asemenea, ar trebui să reţineţi această adresă Web, pentru referinţe viitoare: situl SAX al lui David Megginson, la `http://www.megginson.com/SAX/`. Aici veţi găsi motorul SAX2 în Java, ceva informaţii despre SAX şi alte lucruri utile. Există acolo şi alte motoare analizoare, Lark de la Tim Bray, MSXML de la Microsoft, XML Library de la Sun Microsystems şi SXP de la Silfide. XML Software Site, la `http://www.xmlsoftware.com/`, are legături la toate acestea.

Dar destul cu teoria, haideţi să *facem* ceva!

Tratarea evenimentelor în SAX

Acest capitol este prea mic pentru a studia în amănunt secretele prelucrării SAX şi XML, dar veţi face mai multe în capitolele următoare, iar pentru moment puteţi rula o variaţie la codul Java de pe situl lui David Megginson, la `http://www.megginson.com/SAX/`, care afişează mesaje atunci când găseşte noduri şi când le părăseşte. În primul rând, va trebui să înlocuiţi câteva din handlerele prestabilite pentru `HandlerBase`, un subhandler în `DocumentHandler`, handlerul care recepţionează evenimentele start-element şi end-element, instrucţiuni de prelucrare, date caracter şi câteva lucruri XML de bază:

```
import org.xml.sax.HandlerBase;
import org.xml.sax.AttributeList;
  public class MyTreeTracer extends HandlerBase {
    public void startElement (String name, AttributeList atts) {
      System.out.println("Entering element: " + name);}
    public void endElement (String name) {
      System.out.println("Leaving element: " + name);}
  }
```

Acum trebuie să apelaţi noile noastre handlere:

```
import org.xml.sax.Parser;
import org.xml.sax.DocumentHandler;
import org.xml.sax.helpers.ParserFactory;
  public class SAXApp {
    static final String parserClass = "com.microstar.xml.SAXDriver";
    public static void main (String args[])
      throws Exception {
      Parser parser = ParserFactory.makeParser(parserClass);
      DocumentHandler handler = new MyTreeTracer();
      parser.setDocumentHandler(handler);
      for (int i = 0; i < args.length; i++) {
        parser.parse(args[i]);}
    }
  }
```

Valoarea `"com.microstar.xml.SAXDriver"` este pentru Ælfred, driverul SAX de la Microstar. Dacă utilizaţi o altă versiune, atunci va trebui să modificaţi acest şir astfel încât să se potrivească cu driverul dumneavoastră. Acest cod nu este foarte util, după cum probabil că v-aţi dat seama, dar cu neaşteptat de puţină muncă îl puteţi transforma într-un instrument de aplicaţie efectiv.

Să spunem că doriţi să transformaţi cod XML în XHTML. Veţi mai avea nevoie de câteva extensii la clasa `DocumentHandler`, `StartDocument` (pentru a trimite la ieşire prologul XHTML şi începutul documentului XHTML), `EndDocument` (pentru a încheia documentul nostru) şi `characters`, pentru a transfera caracterele întâlnite. Veţi avea nevoie, de asemenea, de puţin mai multă capacitate I/O:

```
import org.xml.sax.HandlerBase;
import org.xml.sax.AttributeList;
import java.io.*;
  public class MyTreeTracer extends HandlerBase {
    private PrintWriter fout;
    public MyTreeTracer() throws IOException {
      fout = new PrintWriter(new FileWriter("aout.html")); }
    public void startDocument() {
      fout.println("<?xml version='1.0'?>");
      fout.println("<!DOCTYPE html PUBLIC '-//W3C//DTD XHTML 1.0 Strict//EN'");
      fout.println("'http://www.w3.org/TR/xhtml1/DTD/strict.dtd'>");
      fout.println("<html xmlns='http://www.w3.org/TR/xhtml1' />");
      fout.println("<head>");
      fout.println("<title>MyTreeTracer Test</title>");
      fout.println("</head>");
      fout.println("<body>"); }
    public void endDocument() {
      fout.println("</body>");
      fout.println("</html>");
      fout.close(); }
    public void startElement (String name, AttributeList atts) {
      fout.println("<p>");
      fout.print(name + ": "); }
    public void endElement (String name) {
      fout.println(" :" + name);
      fout.println("</p>"); }
    public void characters (char ch[], int start, int length) {
      for (int i=start; i < start+length; i++) {
        fout.print(ch[i]); }
      }
  }
```

Nici acest cod nu face de fapt prea multe, doar preia toate datele caracter şi le afişează pe o linie încadrată de numele elementului care conţine datele. Dar rezultatul este XHTML 1.0 curat, iar prin teste corespunzătoare ale numelor elementelor şi prin prelucrare puteţi transforma cod XML în orice doriţi.

Sugestie de la LeeAnne.com Words to weave by

Găsirea unor şabloane existente în domeniul public care să facă *aproape* exact ceea ce doriţi poate accelera procesul dumneavoastră de învăţare şi timpul necesar dezvoltării până când memoraţi expresiile şi convenţiile limbajelor. Puteţi experimenta cu orice cod, dar înainte de a trece la producţia efectivă folosind ca bază un anumit cod, asiguraţi-vă că este complet disponibil pentru folosirea sa în modul dorit de dumneavoastră, citind foarte atent condiţiile de licenţă şi contactând autorul sau deţinătorul drepturilor de autor dacă aveţi o îndoială cât de mică.

Scrierea handlerelor de evenimente

În secțiunea anterioară, „Tratarea evenimentelor în SAX", ați scris un handler de evenimente practic, utilizând facilitățile unui sistem real, SAX. Această secțiune oferă mai multe detalii, într-un mod puțin mai teoretic.

Părțile dificile ale scrierii handlerelor de evenimente au fost deja realizate de cei (sau cel, în cazul SAX) care au creat analizorul/handlerul de evenimente pe care îl utilizați. Ei furnizează o serie de șabloane de clase, multe dintre ele fiind doar racorduri, care efectuează munca grea de parcurgere a arborelui documentului și de sesizare a evenimentelor. Clasele publice nu fac de obicei prea multe, dar servesc scopul util de a păstra structura funcțională până când completați spațiile goale cu codul dumneavoastră. Deci până când extindeți acele clase cu metode utile, ele nu fac nimic, în timp ce analizorul trece cu veselie peste cod parcurgând ramurile în sus și în jos, lucru făcut de altfel foarte bine.

Ori de câte ori clasele găsesc ceva interesant, cum ar fi un nod sau o etichetă, câteva caractere, o instrucțiune de prelucrare, sau ceva asemănător, ele trimit un mesaj clasei responsabile pentru acea stare interesantă. Deci `startDocument()`, o subclasă din `handlerBase`, este anunțată atunci când analizorul găsește începutul documentului; `endDocument()`, de asemenea o subclasă din `handlerBase`, este anunțată atunci când găsește sfârșitul documentului; iar altele sunt anunțate pentru orice apare între acestea. Deoarece acea clasă este extinsă de dumneavoastră, instanțierile dumneavoastră păstrează toate acele mesaje, deci le puteți utiliza așa cum credeți că este bine. Exact asta s-a întâmplat în exemplu. Deoarece subclasa de bază nu face nimic altceva decât își păstrează locul în analizor, atunci când extindeți metodele disponibile pentru noua ei subclasă, asigurați-vă că face ceva util.

În ciuda faptului că există mult mai multe clase disponibile în Java, fiecare făcând lucruri interesante, conceptul nu este atât de greu după ce scăpați de perspectiva procedurală impusă de limbajele de programare secvențiale. În locul gestionării amănunțite a fiecărui detaliu, le spuneți noilor recruți cum faceți anumite lucruri în programul dumneavoastră și îi lăsați să lucreze. Asemenea unor furnici harnice, ei se vor ocupa de partea lor de treabă fără a vă solicita atenția până când termină. Ei se vor trezi dimineața, vor veni la lucru și se vor duce noaptea acasă, fără a fi nevoie de nici un fel de supraveghere.

SAX în detaliu

Nu există un SAX monolitic care să fie descris în câteva cuvinte sau să fie generat mecanic dintr-un meniu de instrumente predefinite. SAX este o interfață de programare proiectată să fie ușor extensibilă, astfel încât să puteți crea clase și metode noi care să se potrivească precis și eficient operației de efectuat. SAX implementează un set principal de funcții utile în orice situație când se utilizează XML și care lasă proiectarea și implementarea programului în seama dumneavoastră, a programatorului. Nu puteți ignora faptul că pentru a-l utiliza, trebuie într-adevăr să știți cum să scrieți programe.

Totuși, pare potrivită o prezentare a caracteristicilor și proprietăților principale pe care SAX le face disponibile, pentru a ști cu ce puteți lucra înainte de a porni la treabă.

Caracteristici SAX

SAX vine împachetat în două pachete de clase, `org.xml.sax` și `org.xml.sax.helpers`. Primul conține funcționalitatea de bază necesară manevrării documentelor XML, iar cel de-al doilea conține diverse clase de asistență.

Interfețe din `org.xml.sax`

Tabelul 11.1 conține o listă a interfețelor din `org.xml.sax`, împreună cu o scurtă descriere a scopului lor.

Tabelul 11.1 Interfețe din pachetul `org.xml.sax`

Interfață	Scop
`AttributeList`	Nu este recomandată. Această interfață este inclusă pentru compatibilitatea cu versiunile anterioare de SAX. A fost înlocuită de interfața SAX2 `Attributes`, care conține suport pentru zone de nume.
`Attributes`	O interfață pentru listarea atributelor XML.
`ContentHandler`	O interfață pentru obținerea conținutului logic al unui document.
`DocumentHandler`	Nu este recomandată. Această interfață este inclusă pentru compatibilitatea cu versiunile anterioare de SAX. A fost înlocuită de interfața SAX2 `ContentHandler`, care conține suport pentru zone de nume.
`DTDHandler`	O interfață pentru interceptarea evenimentelor fundamentale legate de DTD.
`EntityResolver`	O interfață pentru rezolvarea entităților XML.
`ErrorHandler`	O interfață pentru implementarea handlerelor de erori SAX.
`Locator`	O interfață pentru asocierea unui eveniment SAX la o locație de document.
`Parser`	Nu este recomandată. Această interfață este inclusă pentru compatibilitatea cu versiunile anterioare de SAX. A fost înlocuită de interfața SAX2 `XMLReader`, care conține suport pentru zone de nume.
`XMLFilter`	O interfață pentru filtre XML.
`XMLReader`	O interfață pentru citirea documentelor XML.

Clase din `org.xml.sax`

Tabelul 11.2 conține o listă a claselor de bază din `org.xml.sax`, împreună cu o scurtă descriere a scopului lor.

Tabelul 11.2 Clase din pachetul org.xml.sax

Clasă	Scop
HandlerBase	Nu este recomandată. Această clasă este inclusă pentru compatibilitatea cu versiunile anterioare de SAX. A fost înlocuită de interfața SAX2 DocumentHandler.
InputSource	Această clasă identifică o singură sursă de intrare pentru o entitate XML.

Excepții din org.xml.sax

Tabelul 11.3 conține o listă a excepțiilor din org.xml.sax, împreună cu o scurtă descriere a scopului lor.

Tabelul 11.3 Excepții din pachetul org.xml.sax

Excepție	Scop
SAXException	Clasă de excepții pentru interceptarea erorilor și avertismentelor SAX generale.
SAXNotRecognizedException	Clasă de excepții pentru interceptarea identificatorilor necunoscuți.
SAXNotSupportedException	Clasă de excepții pentru interceptarea operațiilor care nu au suport.
SAXParseException	Clasă de excepții pentru interceptarea erorilor și avertismentelor de analizare XML.

Clase din org.xml.sax.helpers

Tabelul 11.4 conține o listă a claselor din org.xml.sax.helpers împreună cu o scurtă descriere a scopului lor.

Tabelul 11.4 Clase din pachetul org.xml.sax.helpers

Clasă	Scop
AttributeListImpl	Nu este recomandată. Această clasă este inclusă pentru compatibilitatea cu versiunile anterioare de SAX. A fost înlocuită de Attributes din SAX2, care este implementată în clasa de asistență AttributesImpl.
AttributesImpl	Această clasă implementează o versiune prestabilită a interfeței Attributes, care poate fi extinsă dacă este nevoie.
DefaultHandler	Această clasă este o bază prestabilită pentru construirea prin extensie de handlere de evenimente SAX2.
LocatorImpl	Această clasă implementează o versiune prestabilită a lui Locator, care poate fi extinsă dacă este nevoie.
NamespaceSupport	Această clasă implementează logica prestabilită Namespace, pentru a fi utilizată de driverele SAX.
ParserAdapter	Această clasă vă permite să utilizați un Parser SAX1 ca un XMLReader SAX2.

ParserFactory	Nu este recomandată. Această clasă este inclusă pentru compatibilitatea cu versiunile anterioare de SAX. A fost înlocuită de interfața SAX2 Parser.
XMLFilterImpl	Această clasă este o clasă de bază pentru derivarea prin extensie a filtrelor XML.
XMLReaderAdapter	Această clasă vă permite să utilizați un XMLReader SAX2 ca un Parser SAX1.
XMLReaderFactory	Această clasă este utilizată pentru crearea unui XMLReader.

Lista alfabetică a claselor și metodelor SAX

Tabelul 11.5 conține o listă alfabetică a tuturor caracteristicilor din SAX2, împreună cu o sugestie concisă de sintaxă și o descriere scurtă a scopului lor.

Tabelul 11.5 Caracteristici din SAX2

Caracteristică	Scop
addAttribute(*String*, *String*, *String*)	Nu este recomandată – O metodă din clasa org.xml.sax.helpers.AttributeListImpl. Adaugă un atribut la o listă de atribute.
addAttribute(*String*, *String*, *String*, *String*, *String*)	O metodă din clasa org.xml.sax.helpers.AttributesImpl. Adaugă un atribut la sfârșitul unei liste de atribute.
AttributeList	Nu este recomandată – Interfața org.xml.sax.AttributesList. Această interfață a fost înlocuită de interfața SAX2 Attributes, care conține suport pentru zone de nume.
AttributeListImpl	Nu este recomandată – Clasa org.xml.sax.helpers.AttributeListImpl. Această clasă implementează o interfață nerecomandată, AttributeList, care a fost înlocuită de Attributes din SAX2, care este implementată în clasa de asistență AttributesImpl.
AttributeListImpl()	Nu este recomandată – Constructorul clasei org.xml.sax.helpers.AttributeListImpl. Creează o listă de atribute vidă.
AttributeListImpl (AttributeList)	Nu este recomandată – Constructorul clasei org.xml.sax.helpers.AttributeListImpl. Construiește o copie persistentă a unei liste de atribute existente.
Attributes	Interfața org.xml.sax.Attributes. Această interfață furnizează metode care returnează o listă de atribute XML.
AttributesImpl	Clasa org.xml.sax.helpers.AttributesImpl. Această clasă este implementarea prestabilită a interfeței Attributes.

Tabelul 11.5 (continuare)

`AttributesImpl()`	Constructorul clasei `org.xml.sax.helpers.AttributesImpl`. Acest constructor creează un nou obiect vid `AttributesImpl`.
`AttributesImpl(Attributes)`	Constructorul clasei `org.xml.sax.helpers.AttributesImpl`. Creează o copie a unui obiect `Attributes` existent.
`characters(char[], int, int)`	O metodă din interfața `org.xml.sax.ContentHandler`. Interceptează notificări ale datelor caracter.
`characters(char[], int, int)`	Nu este recomandată – O metodă din clasa `org.xml.sax.HandlerBase`. Interceptează notificări ale datelor caracter din interiorul unui element.
`characters(char[], int, int)`	Nu este recomandată – O metodă din interfața `org.xml.sax.DocumentHandler`. Interceptează notificări ale datelor caracter.
`characters(char[], int, int)`	O metodă din clasa `org.xml.sax.helpers.XMLReaderAdapter`. Adaptează un eveniment cu caractere SAX2 la SAX1.
`characters(char[], int, int)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment cu date caracter.
`characters(char[], int, int)`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări ale datelor caracter din interiorul unui element.
`characters(char[], int, int)`	O metodă din clasa `org.xml.sax.helpers.ParserAdapter`. Adaptează un eveniment cu caractere SAX1 la SAX2.
`clear()`	Nu este recomandată – O metodă din clasa `org.xml.sax.helpers.AttributeListImpl`. Șterge o listă de atribute.
`clear()`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Șterge o listă de atribute în vederea reutilizării.
`ContentHandler`	Interfața `org.xml.sax.ContentHandler`. Interceptează notificări ale conținutului logic al unui document.
`createXMLReader()`	O metodă statică din clasa `org.xml.sax.helpers.XMLReaderFactory`. Încearcă să creeze un cititor XML dintr-o proprietate sistem.
`createXMLReader(String)`	O metodă statică din clasa `org.xml.sax.helpers.XMLReaderFactory`. Încearcă să creeze un cititor XML dintr-un nume de clasă.
`declarePrefix(String, String)`	O metodă din clasa `org.xml.sax.helpers.NamespaceSupport`. Declară un prefix `Namespace`.
`DefaultHandler`	Clasa `org.xml.sax.helpers.DefaultHandler`. Clasa de bază prestabilită pentru handlerele de evenimente SAX2.

`DefaultHandler()`	Constructorul clasei `org.xml.sax.helpers.DefaultHandler`.
`DocumentHandler`	Nu este recomandată – Interfața `org.xml.sax.DocumentHandler`. Această interfață a fost înlocuită de interfața SAX2 `ContentHandler`, care conține suport pentru zone de nume.
`DTDHandler`	Interfața `org.xml.sax.DTDHandler`. Interceptează notificări ale evenimentelor fundamentale legate de DTD.
`endDocument()`	O metodă din interfața `org.xml.sax.ContentHandler`. Interceptează notificări despre sfârșitul unui document.
`endDocument()`	Nu este recomandată – O metodă din clasa `org.xml.sax.HandlerBase`. Interceptează notificări despre sfârșitul documentului.
`endDocument()`	Nu este recomandată – O metodă din interfața `org.xml.sax.DocumentHandler`. Interceptează notificări despre sfârșitul unui document.
`endDocument()`	O metodă din clasa `org.xml.sax.helpers.XMLReaderAdapter`. Evenimentul sfârșit de document.
`endDocument()`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment sfârșit de document.
`endDocument()`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări despre sfârșitul documentului.
`endDocument()`	O metodă din clasa `org.xml.sax.helpers.ParserAdapter`. Adaptează un eveniment sfârșit de document SAX1 la SAX2.
`endElement(String)`	Nu este recomandată – O metodă din clasa `org.xml.sax.HandlerBase`. Interceptează notificări despre sfârșitul unui element.
`endElement(String)`	Nu este recomandată – O metodă din interfața `org.xml.sax.DocumentHandler`. Interceptează notificări despre sfârșitul unui element.
`endElement(String)`	O metodă din clasa `org.xml.sax.helpers.ParserAdapter`. Adaptează un eveniment de sfârșit de element SAX1.
`endElement(String, String, String)`	O metodă din interfața `org.xml.sax.ContentHandler`. Interceptează notificări despre sfârșitul unui element.
`endElement(String, String, String)`	O metodă din clasa `org.xml.sax.helpers.XMLReaderAdapter`. Adaptează un eveniment de sfârșit de element SAX2.
`endElement(String, String, String)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment sfârșit de element.
`endElement(String, String, String)`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări despre sfârșitul unui element.

Tabelul 11.5 (continuare)

endPrefixMapping(String)	O metodă din interfața org.xml.sax.ContentHandler. Încheie domeniul de valabilitate al unei mapări prefix-URI.
endPrefixMapping(String)	O metodă din clasa org.xml.sax.helpers. XMLReaderAdapter. Adaptează un eveniment de încheiere a unei mapări prefix SAX2.
endPrefixMapping(String)	O metodă din clasa org.xml.sax.helpers. XMLFilterImpl. Filtrează un eveniment de încheiere a unei mapări prefix Namespace.
endPrefixMapping(String)	O metodă din clasa org.xml.sax.helpers. DefaultHandler. Interceptează notificări despre sfârșitul unei mapări Namespace.
EntityResolver	Interfața org.xml.sax.EntityResolver. Interfață fundamentală pentru rezolvarea entităților.
error(SAXParseException)	Nu este recomandată – O metodă din clasa org.xml. sax.HandlerBase. Interceptează notificări ale unei erori de analiză recuperabile.
error(SAXParseException)	O metodă din interfața org.xml.sax.ErrorHandler. Interceptează notificări ale unei erori recuperabile.
error(SAXParseException)	O metodă din clasa org.xml.sax.helpers. XMLFilterImpl. Filtrează un eveniment de eroare.
error(SAXParseException)	O metodă din clasa org.xml.sax.helpers. DefaultHandler. Interceptează notificări ale unei erori de analiză recuperabile.
ErrorHandler	Interfața org.xml.sax.ErrorHandler. Interfață fundamentală pentru handlerele de erori SAX.
fatalError(SAXParseException)	Nu este recomandată – O metodă din clasa org.xml. sax.HandlerBase. Raportează o eroare fatală de analiză XML.
fatalError(SAXParseException)	O metodă din interfața org.xml.sax.ErrorHandler. Interceptează notificări ale unei erori irecuperabile.
fatalError(SAXParseException)	O metodă din clasa org.xml.sax.helpers. XMLFilterImpl. Filtrează un eveniment de eroare fatală.
fatalError(SAXParseException)	O metodă din clasa org.xml.sax.helpers. DefaultHandler. Raportează o eroare fatală de analiză XML.
getByteStream()	O metodă din clasa org.xml.sax.InputSource. Primește fluxul de octeți pentru această sursă de intrare.
getCharacterStream()	O metodă din clasa org.xml.sax.InputSource. Primește fluxul de caractere pentru această sursă de intrare.
getColumnNumber()	O metodă din interfața org.xml.sax.Locator. Returnează numărul coloanei unde se încheie evenimentul documentului curent.

getColumnNumber()	O metodă din clasa org.xml.sax.SAXParseException. Numărul coloanei de la sfârșitul textului unde a apărut excepția.
getColumnNumber()	O metodă din clasa org.xml.sax.helpers.LocatorImpl. Returnează numărul coloanei salvate (numerotarea începe de la 1).
getDeclaredPrefixes()	O metodă din clasa org.xml.sax.helpers. NamespaceSupport. Returnează o enumerare a tuturor prefixelor declarate în acest context.
getDTDHandler()	O metodă din interfața org.xml.sax.XMLReader. Returnează handlerul DTD-ului curent.
getDTDHandler()	O metodă din clasa org.xml.sax.helpers.XMLFilterImpl. Primește handlerul de evenimente al DTD-ului curent.
getDTDHandler()	O metodă din clasa org.xml.sax.helpers.ParserAdapter. Returnează handlerul DTD-ului curent.
getEncoding()	O metodă din clasa org.xml.sax.InputSource. Primește codificarea caracter pentru un flux de octeți sau un URI.
getEntityResolver()	O metodă din interfața org.xml.sax.XMLReader. Returnează rezolvitorul de entități curent.
getEntityResolver()	O metodă din clasa org.xml.sax.helpers.XMLFilterImpl. Primește rezolvitorul de entități curent.
getEntityResolver()	O metodă din clasa org.xml.sax.helpers.ParserAdapter. Returnează rezolvitorul de entități curent.
getErrorHandler()	O metodă din interfața org.xml.sax.XMLReader. Returnează handlerul de erori curent.
getErrorHandler()	O metodă din clasa org.xml.sax.helpers.XMLFilterImpl. Primește handlerul de erori curent.
getErrorHandler()	O metodă din clasa org.xml.sax.helpers.ParserAdapter. Returnează handlerul de erori curent.
getException()	O metodă din clasa org.xml.sax.SAXException. Returnează excepția înglobată, dacă există.
getFeature(String)	O metodă din interfața org.xml.sax.XMLReader. Caută valoarea unei caracteristici.
getFeature(String)	O metodă din clasa org.xml.sax.helpers.XMLFilterImpl. Caută starea unei caracteristici.
getFeature(String)	O metodă din clasa org.xml.sax.helpers.ParserAdapter. Verifică o caracteristică a unui analizor.
getIndex(String)	O metodă din interfața org.xml.sax.Attributes. Caută indexul unui atribut printr-un nume calificat XML 1.0.
getIndex(String)	O metodă din clasa org.xml.sax.helpers.AttributesImpl. Caută indexul unui atribut printr-un nume calificat pentru zonă de nume.

Tabelul 11.5 (continuare)

`getIndex(String, String)`	O metodă din interfața `org.xml.sax.Attributes`. Caută indexul unui atribut printr-un nume `Namespace`.
`getIndex(String, String)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Caută indexul unui atribut printr-un nume `Namespace`.
`getLength()`	O metodă din interfața `org.xml.sax.Attributes`. Returnează numărul de atribute dintr-o listă.
`getLength()`	Nu este recomandată – O metodă din interfața `org.xml.sax.AttributeList`. Returnează numărul de atribute din această listă.
`getLength()`	Nu este recomandată – O metodă din clasa `org.xml.sax.helpers.AttributeListImpl`. Returnează numărul de atribute din listă.
`getLength()`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Returnează numărul de atribute din listă.
`getLineNumber()`	O metodă din interfața `org.xml.sax.Locator`. Returnează numărul liniei unde se încheie evenimentul documentului curent.
`getLineNumber()`	O metodă din clasa `org.xml.sax.SAXParseException`. Numărul liniei de la sfârșitul textului unde a apărut excepția.
`getLineNumber()`	O metodă din clasa `org.xml.sax.helpers.LocatorImpl`. Returnează numărul liniei salvate (numerotarea începe de la 1).
`getLocalName(int)`	O metodă din interfața `org.xml.sax.Attributes`. Caută după index numele local al unui atribut.
`getLocalName(int)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Returnează numele local al unui atribut.
`getMessage()`	O metodă din clasa `org.xml.sax.SAXException`. Returnează un mesaj detaliat referitor la această excepție.
`getName(int)`	Nu este recomandată – O metodă din interfața `org.xml.sax.AttributeList`. Returnează numele unui atribut după poziția lui în această listă.
`getName(int)`	Nu este recomandată – O metodă din clasa `org.xml.sax.helpers.AttributeListImpl`. Primește numele unui atribut după poziția sa.
`getParent()`	O metodă din interfața `org.xml.sax.XMLFilter`. Primește părintele `XMLReader`.
`getParent()`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Primește părintele `XMLReader`.

`getPrefix(String)`	O metodă din clasa `org.xml.sax.helpers.NamespaceSupport`. Returnează unul dintre prefixele mapate unui `Namespace` URI.
`getPrefixes()`	O metodă din clasa `org.xml.sax.helpers.NamespaceSupport`. Returnează o enumerare a tuturor prefixelor curent declarate.
`getPrefixes(String)`	O metodă din clasa `org.xml.sax.helpers.NamespaceSupport`. Returnează o enumerare a tuturor prefixelor curent declarate pentru un URI.
`getProperty(String)`	O metodă din interfața `org.xml.sax.XMLReader`. Caută valoarea unei proprietăți.
`getProperty(String)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Caută valoarea unei proprietăți.
`getProperty(String)`	O metodă din clasa `org.xml.sax.helpers.ParserAdapter`. Primește o proprietate a analizorului.
`getPublicId()`	O metodă din interfața `org.xml.sax.Locator`. Returnează identificatorul public pentru evenimentul documentului curent.
`getPublicId()`	O metodă din clasa `org.xml.sax.SAXParseException`. Primește identificatorul public al entității în care a apărut excepția.
`getPublicId()`	O metodă din clasa `org.xml.sax.InputSource`. Primește identificatorul public pentru această sursă de intrare.
`getPublicId()`	O metodă din clasa `org.xml.sax.helpers.LocatorImpl`. Returnează identificatorul public salvat.
`getQName(int)`	O metodă din interfața `org.xml.sax.Attributes`. Caută după index numele calificat XML 1.0 al unui atribut.
`getQName(int)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Returnează numele calificat (prefixat) al unui atribut.
`getSystemId()`	O metodă din interfața `org.xml.sax.Locator`. Returnează identificatorul sistem pentru evenimentul documentului curent.
`getSystemId()`	O metodă din clasa `org.xml.sax.SAXParseException`. Primește identificatorul sistem al entității în care a apărut excepția.
`getSystemId()`	O metodă din clasa `org.xml.sax.InputSource`. Primește identificatorul sistem pentru această sursă de intrare.
`getSystemId()`	O metodă din clasa `org.xml.sax.helpers.LocatorImpl`. Returnează identificatorul sistem salvat.
`getType(int)`	O metodă din interfața `org.xml.sax.Attributes`. Caută după index tipul unui atribut.
`getType(int)`	Nu este recomandată – O metodă din interfața `org.xml.sax.AttributeList`. Returnează tipul unui atribut din listă (după poziția sa).

Tabelul 11.5 (continuare)

`getType(int)`	Nu este recomandată – O metodă din clasa `org.xml.sax.helpers.AttributeListImpl`. Primește tipul unui atribut (după poziția sa).
`getType(int)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Returnează după index tipul unui atribut.
`getType(String)`	O metodă din interfața `org.xml.sax.Attributes`.Caută tipul unui atribut după un nume calificat pentru zonă de nume XML 1.0.
`getType(String)`	Nu este recomandată – O metodă din interfața `org.xml.sax.AttributeList`. Returnează după nume tipul unui atribut din listă.
`getType(String)`	Nu este recomandată – O metodă din clasa `org.xml.sax.helpers.AttributeListImpl`. Primește după nume tipul unui atribut.
`getType(String)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Caută tipul unui atribut după numele calificat (prefixat).
`getType(String, String)`	O metodă din interfața `org.xml.sax.Attributes`. Caută tipul unui atribut după un nume `Namespace`.
`getType(String, String)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Caută tipul unui atribut după un nume `Namespace` calificat.
`getURI(int)`	O metodă din interfața `org.xml.sax.Attributes`. Caută după index URI-ul `Namespace` al unui atribut.
`getURI(int)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Returnează URI-ul `Namespace` al unui atribut.
`getURI(String)`	O metodă din clasa `org.xml.sax.helpers.NamespaceSupport`. Caută un prefix și obține URI-ul `Namespace` mapat curent.
`getValue(int)`	O metodă din interfața `org.xml.sax.Attributes`. Caută după index valoarea unui atribut.
`getValue(int)`	Nu este recomandată – O metodă din interfața `org.xml.sax.AttributeList`. Returnează valoarea unui atribut din listă (după poziția sa).
`getValue(int)`	Nu este recomandată – O metodă din clasa `org.xml.sax.helpers.AttributeListImpl`. Primește valoarea unui atribut (după poziția sa).
`getValue(int)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Returnează după index valoarea unui atribut.
`getValue(String)`	O metodă din interfața `org.xml.sax.Attributes`. Caută valoarea unui atribut după un nume calificat XML 1.0.

`getValue(String)`	Nu este recomandată – O metodă din interfața `org.xml.sax.AttributeList`. Returnează valoarea unui atribut din listă (după nume).
`getValue(String)`	Nu este recomandată – O metodă din clasa `org.xml.sax.helpers.AttributeListImpl`. Primește valoarea unui atribut (după nume).
`getValue(String)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Caută valoarea unui atribut după numele calificat (prefixat).
`getValue(String, String)`	O metodă din interfața `org.xml.sax.Attributes`. Caută valoarea unui atribut după un nume `Namespace`.
`getValue(String, String)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Caută valoarea unui atribut după un nume `Namespace` calificat.
`HandlerBase`	Nu este recomandată – Clasa `org.xml.sax.HandlerBase`. Această clasă funcționează cu interfața nerecomandată `DocumentHandler`. A fost înlocuită de clasa SAX2 `DefaultHandler`.
`HandlerBase()`	Nu este recomandată – Constructorul clasei `org.xml.sax.HandlerBase`.
`ignorableWhitespace(char[], int, int)`	O metodă din interfața `org.xml.sax.ContentHandler`. Interceptează notificări despre spațiile albe care pot fi ignorate din conținutul elementelor.
`ignorableWhitespace(char[], int, int)`	Nu este recomandată – O metodă din clasa `org.xml.sax.HandlerBase`. Interceptează notificări despre spațiile albe care pot fi ignorate din conținutul elementelor.
`ignorableWhitespace(char[], int, int)`	Nu este recomandată – O metodă din interfața `org.xml.sax.DocumentHandler`. Interceptează notificări despre spațiile albe care pot fi ignorate din conținutul elementelor.
`ignorableWhitespace(char[], int, int)`	O metodă din clasa `org.xml.sax.helpers.XMLReaderAdapter`. Adaptează un eveniment SAX2, cu spații albe care pot fi ignorate, la SAX1.
`ignorableWhitespace(char[], int, int)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment cu spații albe care pot fi ignorate.
`ignorableWhitespace(char[], int, int)`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări despre spațiile albe care pot fi ignorate din conținutul elementelor.
`ignorableWhitespace(char[], int, int)`	O metodă din clasa `org.xml.sax.helpers.ParserAdapter`. Adaptează un eveniment SAX1, cu spații albe care pot fi ignorate, la SAX2.
`InputSource`	Clasa `org.xml.sax.InputSource`. Identifică o singură sursă de intrare pentru o entitate XML.

Tabelul 11.5 (continuare)

`InputSource()`	Constructor al clasei `org.xml.sax.InputSource`. Constructorul prestabilit, fără nici un argument, pentru o sursă de intrare.
`InputSource(InputStream)`	Constructor al clasei `org.xml.sax.InputSource`. Creează o sursă de intrare nouă având asociat un flux de octeți.
`InputSource(Reader)`	Constructor al clasei `org.xml.sax.InputSource`. Creează o sursă de intrare nouă având asociat un flux de caractere.
`InputSource(String)`	Constructor al clasei `org.xml.sax.InputSource`. Creează o sursă de intrare nouă având asociat un identificator de sistem.
`Locator`	Interfața `org.xml.sax.Locator`. Interfață pentru asocierea unui eveniment SAX cu locația unui document.
`LocatorImpl`	Clasa `org.xml.sax.helpers.LocatorImpl`. Furnizează o implementare comodă, opțională, a lui `Locator`.
`LocatorImpl()`	Constructor al clasei `org.xml.sax.helpers.LocatorImpl`. Constructor fără nici un argument.
`LocatorImpl(Locator)`	Constructor al clasei `org.xml.sax.helpers.LocatorImpl`. Constructor de copiere.
`makeParser()`	Nu este recomandată – O metodă statică din clasa `org.xml.sax.helpers.ParserFactory`. Creează un analizor SAX nou utilizând proprietatea sistem `'org.xml.sax.parser'`.
`makeParser(String)`	Nu este recomandată – O metodă statică din clasa `org.xml.sax.helpers.ParserFactory`. Creează un nou obiect analizor SAX utilizând numele de clasă furnizat.
`NamespaceSupport`	Clasa `org.xml.sax.helpers.NamespaceSupport`. Încapsulează logica `Namespace` pentru a fi utilizată de driverele SAX.
`NamespaceSupport()`	Constructor al clasei `org.xml.sax.helpers.NamespaceSupport`. Creează un nou obiect `NamespaceSupport`.
`notationDecl(String, String, String)`	Nu este recomandată – O metodă din clasa `org.xml.sax.HandlerBase`. Interceptează notificări ale unei declarații de notație.
`notationDecl(String, String, String)`	O metodă din interfața `org.xml.sax.DTDHandler`. Interceptează notificări ale unui eveniment declarație de notație.
`notationDecl(String, String, String)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment declarație de notație.

`notationDecl(String, String, String)`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări ale unei declarații de notație.
`org.xml.sax`	Pachetul `org.xml.sax`.
`org.xml.sax.helpers`	Pachetul `org.xml.sax.helpers`.
`parse(InputSource)`	Nu este recomandată – O metodă din interfața `org.xml.sax.Parser`. Analizează un document XML.
`parse(InputSource)`	O metodă din interfața `org.xml.sax.XMLReader`. Analizează un document XML.
`parse(InputSource)`	O metodă din clasa `org.xml.sax.helpers.XMLReaderAdapter`. Analizează documentul.
`parse(InputSource)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Analizează un document.
`parse(InputSource)`	O metodă din clasa `org.xml.sax.helpers.ParserAdapter`. Analizează un document XML.
`parse(String)`	Nu este recomandată – O metodă din interfața `org.xml.sax.Parser`. Analizează un document XML dintr-un identificator de sistem (URI).
`parse(String)`	O metodă din interfața `org.xml.sax.XMLReader`. Analizează un document XML dintr-un identificator de sistem (URI).
`parse(String)`	O metodă din clasa `org.xml.sax.helpers.XMLReaderAdapter`. Analizează documentul.
`parse(String)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Analizează un document.
`parse(String)`	O metodă din clasa `org.xml.sax.helpers.ParserAdapter`. Analizează un document XML.
`Parser`	Nu este recomandată – Interfața `org.xml.sax.Parser`. Această interfață a fost înlocuită de interfața SAX2 `XMLReader` care conține suport pentru `Namespace`.
`ParserAdapter`	Clasa `org.xml.sax.helpers.ParserAdapter`. Adaptează un `Parser` SAX1 ca un `XMLReader` SAX2.
`ParserAdapter()`	Constructor al clasei `org.xml.sax.helpers.ParserAdapter`. Construiește un nou adaptor analizor.
`ParserAdapter(Parser)`	Constructor al clasei `org.xml.sax.helpers.ParserAdapter`. Construiește un nou adaptor analizor.
`ParserFactory`	Nu este recomandată – Clasa `org.xml.sax.helpers.ParserFactory`. Această clasă funcționează cu interfața `Parser`, care nu este recomandată.
`popContext()`	O metodă din clasa `org.xml.sax.helpers.NamespaceSupport`. Revine la contextul `Namespace` anterior.
`processingInstruction (String, String)`	O metodă din interfața `org.xml.sax.ContentHandler`. Interceptează notificări ale unei instrucțiuni de prelucrare.

Tabelul 11.5 (continuare)

processingInstruction (String, String)	Nu este recomandată – O metodă din clasa org.xml.sax.HandlerBase. Interceptează notificări ale unei instrucțiuni de prelucrare.
processingInstruction (String, String)	Nu este recomandată – O metodă din interfața org.xml.sax.DocumentHandler. Interceptează notificări ale unei instrucțiuni de prelucrare.
processingInstruction (String, String)	O metodă din clasa org.xml.sax.helpers. XMLReaderAdapter. Adaptează un eveniment de instrucțiune de prelucrare SAX2.
processingInstruction (String, String)	O metodă din clasa org.xml.sax.helpers.XMLFilterImpl. Filtrează un eveniment de instrucțiune de prelucrare.
processingInstruction (String, String)	O metodă din clasa org.xml.sax.helpers.DefaultHandler. Interceptează notificări ale unei instrucțiuni de prelucrare.
processingInstruction (String, String)	O metodă din clasa org.xml.sax.helpers.ParserAdapter. Adaptează un eveniment de instrucțiune de prelucrare SAX1.
processName(String, String[], boolean)	O metodă din clasa org.xml.sax.helpers. NamespaceSupport. Prelucrează un nume XML 1.0 brut.
pushContext()	O metodă din clasa org.xml.sax.helpers. NamespaceSupport. Pornește un context Namespace nou.
removeAttribute(int)	O metodă din clasa org.xml.sax.helpers.AttributesImpl. Elimină un atribut din listă.
removeAttribute(String)	Nu este recomandată – O metodă din clasa org.xml.sax.helpers.AttributeListImpl. Elimină un atribut din listă.
reset()	O metodă din clasa org.xml.sax.helpers. NamespaceSupport. Resetează pentru o utilizare nouă obiectul NamespaceSupport.
resolveEntity(String, String)	O metodă din interfața org.xml.sax.EntityResolver. Permite aplicației să rezolve entitățile externe.
resolveEntity(String, String)	Nu este recomandată – O metodă din clasa org.xml.sax.HandlerBase. Rezolvă o entitate externă.
resolveEntity(String, String)	O metodă din clasa org.xml.sax.helpers.XMLFilterImpl. Filtrează rezoluția unei entități externe.
resolveEntity(String, String)	O metodă din clasa org.xml.sax.helpers. DefaultHandler. Rezolvă o entitate externă.
SAXException	Excepția org.xml.sax.SAXException. Încapsulează o eroare generală sau un avertisment general SAX.
SAXException(Exception)	Constructor al clasei org.xml.sax.SAXException. Creează o SAXException nouă, înfășurând o excepție existentă.

SAXException(String)	Constructor al clasei org.xml.sax.SAXException. Creează o SAXException nouă.
SAXException(String, Exception)	Constructor al clasei org.xml.sax.SAXException. Creează o SAXException nouă dintr-o excepție existentă.
SAXNotRecognizedException	Excepția org.xml.sax.SAXNotRecognizedException. Clasă de excepții pentru un identificator necunoscut.
SAXNotRecognizedException (String)	Constructor al clasei org.xml.sax. SAXNotRecognizedException. Construiește o excepție nouă, cu un anumit mesaj.
SAXNotSupportedException	Excepția org.xml.sax.SAXNotSupportedException. Clasă de excepții pentru o operație care nu este permisă.
SAXNotSupportedException (String)	Constructor al clasei org.xml.sax. SAXNotSupportedException. Construiește o excepție nouă, cu un anumit mesaj.
SAXParseException	Excepția org.xml.sax.SAXParseException. Încapsulează o eroare sau un avertisment de analiză XML.
SAXParseException(String, Locator)	Constructor al clasei org.xml.sax. SAXParseException. Creează o SAXParseException nouă dintr-un mesaj și un Locator.
SAXParseException(String, Locator, Exception)	Constructor al clasei org.xml.sax.SAXParseException. Înfășoară o excepție existentă într-o SAXParseException.
SAXParseException(String, String, String, int, int)	Constructor al clasei org.xml.sax.SAXParseException. Creează o SAXParseException nouă.
SAXParseException(String, String, String, int, int, Exception)	Constructor al clasei org.xml.sax.SAXParseException. Creează o SAXParseException nouă, cu o excepție înglobată.
setAttribute(int, String, String, String, String)	O metodă din clasa org.xml.sax.helpers.AttributesImpl. Setează un atribut dintr-o listă de atribute.
setAttributeList(AttributeList)	Nu este recomandată – O metodă din clasa org.xml.sax.helpers.AttributeListImpl. Setează o întreagă listă de atribute, eliminând conținutul anterior.
setAttributes(Attributes)	O metodă din clasa org.xml.sax.helpers. AttributesImpl. Copiază un întreg obiect Attributes.
setByteStream(InputStream)	O metodă din clasa org.xml.sax.InputSource. Setează fluxul de octeți pentru această sursă de intrare.
setCharacterStream(Reader)	O metodă din clasa org.xml.sax.InputSource. Setează fluxul de caractere pentru această sursă de intrare.
setColumnNumber(int)	O metodă din clasa org.xml.sax.helpers.LocatorImpl. Setează numărul coloanei pentru acest Locator (numerotarea începe de la 1).
setContentHandler (ContentHandler)	O metodă din interfața org.xml.sax.XMLReader. Permite unei aplicații să înregistreze un handler de evenimente de conținut.

Tabelul 11.5 (continuare)

setContentHandler (ContentHandler)	O metodă din clasa org.xml.sax.helpers. XMLFilterImpl. Setează handlerul de evenimente de conținut.
setContentHandler (ContentHandler)	O metodă din clasa org.xml.sax.helpers. ParserAdapter. Setează handlerul de conținut.
setDocumentHandler (DocumentHandler)	Nu este recomandată – O metodă din interfața org.xml.sax.Parser. Permite unei aplicații să înregistreze un handler de evenimente de document.
setDocumentHandler (DocumentHandler)	O metodă din clasa org.xml.sax.helpers. XMLReaderAdapter. Înregistrează handlerul de evenimente de document SAX1.
setDocumentLocator(Locator)	O metodă din interfața org.xml.sax.ContentHandler. Interceptează un obiect care localizează originea evenimentelor de document SAX.
setDocumentLocator(Locator)	Nu este recomandată – O metodă din clasa org.xml. sax.HandlerBase. Interceptează un obiect Locator pentru evenimente de document.
setDocumentLocator(Locator)	Nu este recomandată – O metodă din interfața org.xml.sax.DocumentHandler. Interceptează un obiect pentru localizarea originii evenimentelor document SAX.
setDocumentLocator(Locator)	O metodă din clasa org.xml.sax.helpers. XMLReaderAdapter. Setează un Locator document.
setDocumentLocator(Locator)	O metodă din clasa org.xml.sax.helpers. XMLFilterImpl. Filtrează un eveniment Locator document nou.
setDocumentLocator(Locator)	O metodă din clasa org.xml.sax.helpers. DefaultHandler. Interceptează un obiect Locator pentru evenimente document.
setDocumentLocator(Locator)	O metodă din clasa org.xml.sax.helpers. ParserAdapter. Adaptează un eveniment Locator document SAX1 la SAX2.
setDTDHandler(DTDHandler)	Nu este recomandată – O metodă din interfața org.xml.sax.Parser. Permite unei aplicații să înregistreze un handler de evenimente DTD.
setDTDHandler(DTDHandler)	O metodă din interfața org.xml.sax.XMLReader. Permite unei aplicații să înregistreze un handler de evenimente DTD.
setDTDHandler(DTDHandler)	O metodă din clasa org.xml.sax.helpers. XMLReaderAdapter. Înregistrează un handler de evenimente DTD.
setDTDHandler(DTDHandler)	O metodă din clasa org.xml.sax.helpers. XMLFilterImpl. Setează un handler DTD.

setDTDHandler(DTDHandler)	O metodă din clasa org.xml.sax.helpers. ParserAdapter. Setează un handler DTD.
setEncoding(String)	O metodă din clasa org.xml.sax.InputSource. Setează codificarea de caractere, dacă este cunoscută.
setEntityResolver (EntityResolver)	Nu este recomandată – O metodă din interfața org.xml.sax.Parser. Permite unei aplicații să înregistreze un rezolvitor de entitate personalizat.
setEntityResolver (EntityResolver)	O metodă din interfața org.xml.sax.XMLReader. Permite unei aplicații să înregistreze un rezolvitor de entitate.
setEntityResolver (EntityResolver)	O metodă din clasa org.xml.sax.helpers. XMLReaderAdapter. Înregistrează un rezolvitor de entitate.
setEntityResolver (EntityResolver)	O metodă din clasa org.xml.sax.helpers. XMLFilterImpl. Setează un rezolvitor de entitate.
setEntityResolver (EntityResolver)	O metodă din clasa org.xml.sax.helpers.ParserAdapter. Setează un rezolvitor de entitate.
setErrorHandler(ErrorHandler)	Nu este recomandată – O metodă din interfața org.xml.sax.Parser. Permite unei aplicații să înregistreze un handler de eveniment eroare.
setErrorHandler(ErrorHandler)	O metodă din interfața org.xml.sax.XMLReader. Permite unei aplicații să înregistreze un handler de eveniment eroare.
setErrorHandler(ErrorHandler)	O metodă din clasa org.xml.sax.helpers. XMLReaderAdapter. Înregistrează un handler de eveniment eroare.
setErrorHandler(ErrorHandler)	O metodă din clasa org.xml.sax.helpers. XMLFilterImpl. Setează un handler de eveniment eroare.
setErrorHandler(ErrorHandler)	O metodă din clasa org.xml.sax.helpers. ParserAdapter. Setează un handler de eroare.
setFeature(String, boolean)	O metodă din interfața org.xml.sax.XMLReader. Setează starea unei caracteristici.
setFeature(String, boolean)	O metodă din clasa org.xml.sax.helpers. XMLFilterImpl. Setează starea unei caracteristici.
setFeature(String, boolean)	O metodă din clasa org.xml.sax.helpers. ParserAdapter. Setează o caracteristică pentru analizor.
setLineNumber(int)	O metodă din clasa org.xml.sax.helpers.LocatorImpl. Setează numărul liniei pentru acest Locator (numerotarea începe de la 1).
setLocale(Locale)	Nu este recomandată – O metodă din interfața org.xml.sax.Parser. Permite unei aplicații să ceară un Locale pentru erori și avertismente.
setLocale(Locale)	O metodă din clasa org.xml.sax.helpers. XMLReaderAdapter. Setează Locale pentru raportarea de erori.

Tabelul 11.5 (continuare)

`setLocalName(int, String)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Setează numele local al unui anumit atribut.
`setParent(XMLReader)`	O metodă din interfața `org.xml.sax.XMLFilter`. Setează cititorul părinte.
`setParent(XMLReader)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Setează cititorul părinte.
`setProperty(String, object)`	O metodă din interfața `org.xml.sax.XMLReader`. Setează valoarea unei proprietăți.
`setProperty(String, object)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Setează valoarea unei proprietăți.
`setProperty(String, object)`	O metodă din clasa `org.xml.sax.helpers.ParserAdapter`. Setează o proprietate a analizorului.
`setPublicId(String)`	O metodă din clasa `org.xml.sax.InputSource`. Setează identificatorul public pentru această sursă de intrare.
`setPublicId(String)`	O metodă din clasa `org.xml.sax.helpers.LocatorImpl`. Setează identificatorul public pentru acest `Locator`.
`setQName(int, String)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Setează numele calificat al unui anumit atribut.
`setSystemId(String)`	O metodă din clasa `org.xml.sax.InputSource`. Setează identificatorul de sistem pentru această sursă de intrare.
`setSystemId(String)`	O metodă din clasa `org.xml.sax.helpers.LocatorImpl`. Setează identificatorul de sistem pentru acest `Locator`.
`setType(int, String)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Setează tipul unui anumit atribut.
`setURI(int, String)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Setează URI-ul `Namespace` al unui anumit atribut.
`setValue(int, String)`	O metodă din clasa `org.xml.sax.helpers.AttributesImpl`. Setează valoarea unui anumit atribut.
`skippedEntity(String)`	O metodă din interfața `org.xml.sax.ContentHandler`. Interceptează notificări despre o entitate peste care s-a trecut.
`skippedEntity(String)`	O metodă din clasa `org.xml.sax.helpers.XMLReaderAdapter`. Adaptează un eveniment SAX2 de entitate peste care s-a trecut.
`skippedEntity(String)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment de entitate peste care s-a trecut.
`skippedEntity(String)`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări despre o entitate peste care s-a trecut.
`startDocument()`	O metodă din interfața `org.xml.sax.ContentHandler`. Interceptează notificări despre începutul unui document.

`startDocument()`	Nu este recomandată – O metodă din clasa `org.xml.sax.HandlerBase`. Interceptează notificări despre începutul documentului.
`startDocument()`	Nu este recomandată – O metodă din interfața `org.xml.sax.DocumentHandler`. Interceptează notificări despre începutul unui document.
`startDocument()`	O metodă din clasa `org.xml.sax.helpers.XMLReaderAdapter`. Pornește evenimentul documentului.
`startDocument()`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment început de document.
`startDocument()`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări despre începutul documentului.
`startDocument()`	O metodă din clasa `org.xml.sax.helpers.ParserAdapter`. Adaptează un eveniment început de document SAX1.
`startElement(String, AttributeList)`	Nu este recomandată – O metodă din clasa `org.xml.sax.HandlerBase`. Interceptează notificări despre începutul unui element.
`startElement(String, AttributeList)`	Nu este recomandată – O metodă din interfața `org.xml.sax.DocumentHandler`. Interceptează notificări despre începutul unui element.
`startElement(String, AttributeList)`	O metodă din clasa `org.xml.sax.helpers.ParserAdapter`. Adaptează un eveniment `startElement` SAX1.
`startElement(String, String, String, Attributes)`	O metodă din interfața `org.xml.sax.ContentHandler`. Interceptează notificări despre începutul unui element.
`startElement(String, String, String, Attributes)`	O metodă din clasa `org.xml.sax.helpers.XMLReaderAdapter`. Adaptează un eveniment `startElement` SAX2.
`startElement(String, String, String, Attributes)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment `startElement`.
`startElement(String, String, String, Attributes)`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări despre începutul unui element.
`startPrefixMapping (String, String)`	O metodă din interfața `org.xml.sax.ContentHandler`. Începe domeniul de valabilitate al unei mapări `Namespace` cu prefix-URI.
`startPrefixMapping (String, String)`	O metodă din clasa `org.xml.sax.helpers.XMLReaderAdapter`. Adaptează un eveniment `startPrefixMapping` SAX2.
`startPrefixMapping (String, String)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment de început de mapare cu prefix `Namespace`.
`startPrefixMapping (String, String)`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări despre începutul unei mapări `Namespace`.

Tabelul 11.5 (continuare)

`toString()`	O metodă din clasa `org.xml.sax.SAXException`. Suprascrie `toString` pentru a alege orice excepție înglobată.
`unparsedEntityDecl(String, String, String, String)`	Nu este recomandată – O metodă din clasa `org.xml.sax.HandlerBase`. Interceptează notificări despre declarația unei entități neanalizate.
`unparsedEntityDecl(String, String, String, String)`	O metodă din interfața `org.xml.sax.DTDHandler`. Interceptează notificări despre un eveniment de declarare a unei entități neanalizate.
`unparsedEntityDecl(String, String, String, String)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment de declarare a unei entități neanalizate.
`unparsedEntityDecl(String, String, String, String)`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări despre declarația unei entități neanalizate.
`warning(SAXParseException)`	Nu este recomandată – O metodă din clasa `org.xml.sax.HandlerBase`. Interceptează notificări despre un avertisment al analizorului.
`warning(SAXParseException)`	O metodă din interfața `org.xml.sax.ErrorHandler`. Interceptează notificări despre un avertisment sau o eroare.
`warning(SAXParseException)`	O metodă din clasa `org.xml.sax.helpers.XMLFilterImpl`. Filtrează un eveniment de avertizare.
`warning(SAXParseException)`	O metodă din clasa `org.xml.sax.helpers.DefaultHandler`. Interceptează notificări despre un avertisment al analizorului.
`XMLFilter`	Interfața `org.xml.sax.XMLFilter`. Interfață pentru un filtru XML.
`XMLFilterImpl`	Clasa `org.xml.sax.helpers.XMLFilterImpl`. Clasa de bază pentru derivarea prin extensie a unui filtru XML.
`XMLFilterImpl()`	Constructor al clasei `org.xml.sax.helpers.XMLFilterImpl`. Construiește un filtru XML vid,fără părinte.
`XMLFilterImpl(XMLReader)`	Constructor al clasei `org.xml.sax.helpers.XMLFilterImpl`. Construiește un filtru XML cu părintele specificat.
`XMLNS`	Variabila statică zonă de nume din clasa `org.xml.sax.helpers.NamespaceSupport`. Namespace XML este o constantă.
`XMLReader`	Interfața `org.xml.sax.XMLReader`. Interfață pentru citirea unui document XML.
`XMLReaderAdapter`	Clasa `org.xml.sax.helpers.XMLReaderAdapter`. Adaptează un `XMLReader` SAX2 ca `Parser` SAX1.

`XMLReaderAdapter()`	Constructor al clasei `org.xml.sax.helpers.XMLReaderAdapter`. Creează un adaptor nou.
`XMLReaderAdapter(XMLReader)`	Constructor al clasei `org.xml.sax.helpers.XMLReaderAdapter`. Creează un adaptor nou care are asociat un anumit flux de intrare.
`XMLReaderFactory`	Clasa `org.xml.sax.helpers.XMLReaderFactory`. Un `Factory` pentru crearea unui cititor XML.

Tehnici orientate spre obiecte

Ce ați făcut în secțiunea „Tratarea evenimentelor în SAX" era programare orientată spre obiecte. Nu trebuia să știți prea multe despre cum funcționează acele lucruri deoarece prelucrarea internă a fost ascunsă față de dumneavoastră de către clasele pe care le-ați folosit. Puteați lua numai piesele de care aveați nevoie și să le ajustați puțin, iar clasele urmau să vă dea pe tavă tot ce mai era necesar. În lumea O-O acest lucru se numește *moștenire*. Cuvântul este foarte potrivit, deoarece nu trebuie să munciți pentru ea.

Evident, cunoașterea pieselor care trebuie alese este o adevărată scamatorie. Există munți de documentație despre Java, SAX, și despre diferite add-on-uri Java care vă permit să faceți lucruri *chiar mai* interesante, precum și despre SAXON care adaugă multe lucruri utile la API-ul SAX. Nimeni nu le cunoaște pe toate, ci numai acele părți pe care le utilizează mai des. După ce începeți să faceți o muncă adevărată, veți dori să le studiați și să extrageți din marea masă a lor pe cele care par mai utile pentru genul de lucruri pe care doriți să le faceți. Dacă doriți să faceți un instrument XML vizual, vă puteți uita la Swing, biblioteca GUI din Java. Și nu uitați de SAXON, la `http://home.iclweb.com/icl2/mhkay/`.

Și Python este orientat spre obiecte, iar dacă îl cunoașteți deja, este un instrument bun pentru prelucrarea de texte, mai rapid decât Perl și, după părerea mea, puțin mai prietenos. Expunerea Python pentru informarea dumneavoastră este tot ce putem face aici. Dacă vă interesează, situl Python menționat anterior conține legături la entuziaști și la resurse Python, nu neapărat în această ordine.

Un *obiect* este o metaforă a unui lucru real (cel puțin din punct de vedere teoretic) și este un container pentru informațiile despre acel lucru și pentru metodele care prezintă lumii acele informații. El expune doar atât din informațiile sale cât să vă permită să accesați ceea ce aveți nevoie din el și ascunde tot restul. Aceasta înseamnă că nu sunteți ademenit să vă jucați cu biți și pointeri și să faceți alte lucruri periculoase, lucru care vă va scuti mult timp de depanare.

Partea cea mai bună a obiectelor este că ele conțin și informații despre cum să se inițializeze singure atunci când sunt apelate și cum să pună totul deoparte atunci când și-au încheiat lucrul. În terminologia orientată spre obiecte, acestea se numesc *constructorul* și *destructorul* obiectului. În termenii limbajului de asamblare, poartă numele de salvarea și restaurarea regiștrilor. Oricum, este un lucru foarte comod care a fost făcut pentru dumneavoastră. În programare se fac foarte multe greșeli tocmai la acest punct. În Java, un destructor se numește *finalizor*.

Sugestie de la LeeAnne.com Words to weave by

Deşi constructorii şi destructorii sunt de obicei invizibili, şi ei pot fi moşteniţi şi extinşi,ca toate celelalte. În ciuda faptului că acest lucru poate să nu pară important acum, pe măsură ce veţi cunoaşte mai bine Java veţi descoperi că este nevoie să adăugaţi articole care trebuie să fie alocate sau iniţializate la intrare şi eliberate la ieşire.

Acesta nu este un text despre programarea orientată spre obiecte, dar aceasta nu este chiar atât de dificilă. Oamenii înţeleg foarte bine metaforele, iar programarea O-O are ca metaforă un grup de indivizi care cooperează, fiecare dintre ei fiind specializat în anumite operaţii şi neamestecându-se în treburile celorlalţi.

Pentru a construi o casă, aveţi nevoie de betonişti pentru fundaţie, zidari pentru pereţi, instalatori şi electricieni pentru instalaţii, zugravi pentru pereţii interiori şi dulgheri care, făcând acoperişul, să pună capac la toate. În plus, aveţi nevoie de un antreprenor care să gestioneze toate aceste operaţii. Nici unul dintre aceşti indivizi nu trebuie să ştie nimic despre cum se face munca celorlalţi, dar trebuie să cunoască totul despre propria lor muncă; ei trebuie să îşi îndeplinească sarcinile până în acel stadiu în care pot să transfere responsabilitatea altcuiva. Antreprenorul planifică lucrările, dar nu execută nici una dintre ele.

Programarea orientată spre obiecte îmbunătăţeşte această metaforă prin faptul că puteţi clona, la modul figurat, un dulgher, de exemplu, să îi adăugaţi nişte cunoştinţe referitoare la zidărie şi să puneţi dulgherul să se ocupe şi de această operaţie, fără a uita însă ceea ce face un dulgher, deci nu va avea probleme cu cofrajele şi schelăria. Mai mult, puteţi îmbina un dulgher cu un electrician, produsul rezultat cunoscând ambele meserii. Deşi astfel forţăm puţin metafora, moştenirea multiplă face tocmai acest lucru. În primul exemplu era vorba de moştenire simplă.

Clase

Pe scurt, o *clasă* este un prototip care vă permite să construiţi obiecte atunci când este nevoie de ele. Atunci când creaţi un obiect nou, definiţi de fapt clasa sa. Apoi, o instanţă a clasei o instanţiaţi prin crearea unui obiect care utilizează acea clasă ca prototip. Nu puteţi utiliza o clasă dacă nu o instanţiaţi. Clasele sunt concepte foarte puternice, deoarece permit crearea şi utilizarea unui număr nelimitat de obiecte fără a aloca spaţiu de stocare şi fără a manevra alte mecanisme.

În Java, pentru a instanţia o instanţă a unei clase introduceţi codul următor, înlocuind numele în mod corespunzător:

```
classname nume;
nume = new classname();
```

Toate detaliile încurcate sunt ascunse privirii dumneavoastră şi vă puteţi concentra asupra utilizării noului dumneavoastră obiect fără a vă gândi cum să îl construiţi pornind de la zero.

Moştenirea

După cum am sugerat în metafora anterioară, moştenirea este un mod prin care pot fi create clase care să conţină deja majoritatea aptitudinilor lor, astfel încât să poată să îşi dea seama cum să execute noile operaţii. Tot ce aveţi de făcut este să decideţi ce fel de metode suplimentare doriţi să cunoască obiectul şi să codaţi metodele noi. Toate metodele vechi, oricare ar fi acestea, vor funcţiona, dar vor fi disponibile şi metodele noi.

Notă În cel mai general model de orientare spre obiecte, este posibil ca un obiect să moştenească proprietăţi de la mai mulţi strămoşi, deci s-ar putea specifica din punct de vedere teoretic un obiect cu clasele strămoş `ceas` şi `radioReceptor` şi să se creeze o clasă nouă, `ceasRadio`, având proprietăţile amândurora. Totuşi, apar probleme atunci când se încearcă combinarea în acest mod a unor clase înrudite, deci `ceasAnalog` şi `ceasDigital` pot avea metode diferite purtând acelaşi nume, sau pot avea metode identice, dar cu nume diferite. Problemele unei noi clase, `ceasModDual`, descendentă din `ceasAnalog` şi `ceasDigital`, va trebui să fie rezolvate automat de către mediu. Java implementează numai moştenirea simplă, deoarece este cea mai simplă metodă de moştenire şi provoacă şi cele mai puţine probleme. Urmărirea metodelor incluse într-un obiect care are mai mulţi strămoşi este mult prea dificilă pentru cei care nu au puteri supranaturale.

Codul pentru a face acest lucru în Java este la fel de simplu:

```
class newclassname extends classname {
  public void newMethod() {
    orice dorim sa faca noua metoda...;
    }
  }
```

Deoarece extindeţi o clasă preexistentă, tot ce făcea clasa veche este încă posibil şi tot ce trebuie să furnizaţi este funcţionalitatea nouă. Acest lucru asigură că noul cod are o bază solidă pentru un comportament pe deplin înţeles, ceea ca face acceptarea lui mult mai simplă. Clasele care comunică deja cu clasa de bază pot fi extinse pentru a încorpora metoda nouă fără a risca crearea unei interfeţe complet noi.

Java şi Python

Java este unul dintre cele mai de succes limbaje orientate spre obiecte. Înainte de Java, au fost multe încercări de a crea un limbaj orientat spre obiecte care să fie util în lumea reală. Dar erau atât de lente şi de ineficiente încât nu au reuşit niciodată să ajungă în prim plan. Cine îşi aminteşte acum de Eiffel? De Simula? Ei bine, poate vă amintiţi de SmallTalk, dar acesta era un fel de jucărie Xerox. C++ a fost unul din primele succese, deoarece şi-a făcut un capital din baza imensă de programatori C disponibili, dar C++ nu este de fapt un limbaj O-O deoarece nu impune tehnicile O-O. Puteţi utiliza fără probleme C++ pentru a scrie cod C simplu şi să utilizaţi numai C simplu pentru a intra oriunde doriţi. Deşi acest lucru poate fi o alinare pentru ego-ul programatorilor care se bazează numai pe experienţă, nu este orientare spre obiecte.

Java este în parte reclamă, iar în parte realitate, asemenea majorităţii lucrurilor din lumea comercială. Şi înainte au apărut şi au dispărut limbaje universale. Dar Java avea în spatele său puterea unei companii foarte populare şi a sosit şi la momentul potrivit. Internetul a făcut din calculatorul neconectat la el un spaţiu mult prea îngust pentru a vă capta interesul pentru mult timp. O dată cu Internetul au apărut şi o serie de probleme cu interfeţele brevetate, care anterior puteau fi ignorate deoarece rareori ne aventuram prin reţea în afara calculatorului.

Python este şi el orientat spre obiecte, dar este de fapt un limbaj de interpretare destinat realizării de script-uri. Pentru dezvoltatori, avantajul limbajelor de scriptare este că nu există compilare. Scrieţi codul şi îl rulaţi. Aceasta accelerează destul de mult procesul de

dezvoltare. În afară de Python există și alte limbaje de scriptare. Probabil cunoașteți deja JavaScript (ECMAScript) și poate v-ați întâlnit deja cu Perl, care este foarte des utilizat pe Web ca handler CGI. Tcl este un alt limbaj de scriptare cu un număr de susținători. Nu pot să spun că unul sau altul este mai bun, deoarece toate au limitările lor.

Chiar și JavaScript, care este limitat în special la realizarea de pagini Web, vă permite să scrieți fără prea multe probleme programe scurte. Totuși, pentru lucruri importante, Python, Tcl și Perl sunt probabil cele mai bune. Pe de altă parte, dintre aceste trei limbaje, Python este cel mai orientat spre obiecte, deci acesta îmi place cel mai mult. Este o părere personală.

Depanarea SAX

Codul meu nu funcționează!

Ați creat un utilitar minunat pentru a traduce în franceză textul dumneavoastră XML, dar el ignoră întregul text și produce doar o serie de linii albe.

Probabil că nu citiți și nu prelucrați caracterele care utilizează un handler de evenimente diferit. Deoarece vedeți linii albe, probabil că prelucrați corect elementele, presupunând că trimiteți spre ieșire caractere de sfârșit de linie undeva în acea porțiune a codului.

Codul meu a lansat o excepție

Codul dumneavoastră rulează un timp și apoi lansează o excepție care „împodobește" ecranul cu un mesaj iritant.

Încercați să citiți excepția pentru a avea un indiciu despre proveniența ei. Dacă nu reușiți, încercați să parcurgeți codul pentru a asocia erorii un anumit eveniment. Este posibil ca acest lucru să nu funcționeze. Dacă toate încercările eșuează, documentați codul cu mesaje de ieșire în locurile probabile și vedeți dacă vă ajută să restrângeți domeniul de căutare a erorii.

CSS1, CSS2, DSSSL, XSL

CAPITOLUL 12

În acest capitol

Limbaje de stil
Alegerea unui limbaj de stil
Suport pentru limbaje de stil
Combinarea XHTML și XML pentru prelucrări de stil
Utilizarea în tandem a CSS și XSL
Concluzii
Depanarea CSS
Depanarea foilor de stil XSL

Limbaje de stil

Facilitățile simple de machetare și formatare încorporate în standardul HTML (și XHTML) nu sunt suficiente pentru mediul internațional. Ele sunt ostile cu utilizatorul, deoarece nu pot fi suprascrise de utilizatori care au nevoi particulare și pentru că sunt limitate la domeniul lor de valabilitate. De asemenea, ele nu sunt flexibile, obligând utilizatorii documentului să păstreze mai multe copii ale fiecărui document, pentru medii de ieșire diferite. Deși multe pagini, dacă nu cele mai multe, utilizează încă instrumente ostile cu utilizatorul, cum ar fi tabele, spații gif „invizibile" sau comenzi `<font>`, pentru a obține un anumit aspect, această utilizare este lipsită de înțelepciune. În ciuda faptului că unele persoane spun că numărul de situri care utilizează aceste instrumente reprezintă o dovadă că acele instrumente sunt încă importante, este la fel de ușor să spunem că timpul de încărcare jenant și inaccesibilitatea completă a multor pagini Web, dacă nu a majorității, reprezintă o dovadă că a venit momentul unei schimbări.

Acum este posibil să formatați o pagină utilizând comenzi HTML simple care îi permit să rămână vizibilă și utilizabilă în toate browserele, inclusiv în cele utilizate de persoane cu handicap de vedere și în asistenții digitali personali cu aptitudini Web existenți, fără a renunța la modificări de font și la un design grafic interesant. Caracteristicile din CSS, deși nu sunt suportate prea bine de browserele mai vechi, pot fi utilizate, chiar dacă necesită o atenție deosebită, pentru a face tot ceea ce făceau tabelele și etichetele `font` fără a întrerupe sesiunile unui număr semnificativ de utilizatori.

Sugestie de la LeeAnne.com Words to weave by

Pentru mai multe informaţii despre acest subiect, vedeţi articolul „<FONT FACE> Considered Harmful" (<FONT FACE> considerat dăunător), la `http://isoc.bilkent.edu.tr/web_ml/html/fontface.html`, şi articolele lui David Siegel de la `http://www.dsiegel.com/balkanization/` şi `http://www.xml.com/pub/w3j/s1.people.html`.

Lăsând la o parte DSSSL (*Document Style Semantics and Specification Language – limbajul de semantică şi specificare a stilului documentului*), FOSI (*Formatting Output Specification Instance – instanţa de specificare a formatării ieşirii*) şi alte instrumente SGML, CSS1 (*Cascading Style Sheets Level 1 – foi de stil în cascadă de nivelul 1*) a fost prima metodă practică de separare a datelor de modul în care sunt acestea prezentate pe Web. CSS oferea un echilibru de putere între autorul unui sit Web şi utilizator şi a reprezentat măcar un punct de pornire în transformarea documentelor astfel încât să fie suficient de adaptabile pentru a suporta mai multe medii de ieşire. Multe dintre efectele destul de obişnuite acum pe Web, casete de text colorate *sigur*, modificări de font *sigure*, poziţionarea absolută a textului şi imaginilor *fără* improvizaţii de gif-uri de un pixel şi multe altele, au devenit posibile datorită foilor de stil CSS1.

Totuşi, pe măsură ce proiectanţii au lucrat cu CSS1, s-au descoperit alte posibilităţi, CSS2 (Level 2) apărând în scurt timp şi adăugând un suport mai mare pentru tradiţiile tipografice şi de publicare tipărită, suportând mai multe pseudo-clase de formatare şi adăugând câteva opţiuni de internaţionalizare, precum şi suport pentru browserele auditive.

Aceasta este starea actuală a CSS şi promite să se îmbunătăţească şi mai mult pe măsura trecerii timpului. Deci pentru operaţii de formatare relativ simple, care nu necesită transformări complexe ale documentului sursă, CSS rămâne instrumentul ideal pentru proiectanţii Web.

Sugestie de la LeeAnne.com Words to weave by

Deşi CSS poate fi greu de utilizat în mediul browserelor actuale, el este cel mai flexibil şi mai puternic mecanism disponibil pentru a conferi stil documentelor dumneavoastră. Cu grijă şi printr-o utilizare raţională a script-urilor, chiar şi implementările actuale de browsere pot fi determinate să facă lucruri foarte inteligente. Pentru un exemplu excelent de sit nonprofit, vizitaţi `http://www.aegis.com/` care utilizează BrowserHawk, un echivalent script server-side pentru depistarea browserului utilizatorului şi furnizarea dinamică de conţinut diferit în funcţie de ceea ce găseşte.

Totuşi, în realitate, CSS nu rezolvă o întreagă clasă de probleme. Dacă un document necesită un cuprins, dacă o listă ar trebui sortată înaintea afişării sau dacă limba utilizată necesită mai mult decât tehnici de machetare simple, în momentul actual CSS nu poate realiza aceste operaţii. Iar CSS utilizează un limbaj şi o sintaxă complet diferite de HTML, deci atunci când o foaie de stil este plasată in-line, ea trebuie inclusă cu atenţie într-o secvenţă escape prin intermediul comentariilor, pentru a evita blocarea multor browsere mai vechi. Aceasta este o încălcare a filozofiei XML.

Pe lângă toate acestea, implementările actuale ale ambelor niveluri de CSS conţin erori şi nu sunt unitare ca posibilităţi. CSS este un mecanism eficient şi robust pentru realizarea unor operaţii limitate de formatare a afişării, iar pe măsură ce implementările se vor maturiza, va deveni chiar mai valoros. Totuşi, există o nevoie reală pentru mai mult decât poate oferi CSS.

XSL (*Extensible Stylesheet Language – limbajul extensibil al foilor de stil*) este un limbaj de stil reconstituit după tiparul XML, copiind funcţionalitatea lui CSS2 şi adăugând aptitudini de formatare şi transformare chiar mai puternice. XSL şi XSLT, *XSL Transformations* (Transformări XSL), limbajul înrudit de transformare a documentelor, reprezintă punctul culminant al unei lungi evoluţii a standardelor Web. Deoarece XSL are un DTD, nu există nici o scuză pentru eşecul implementării unor porţiuni din specificaţie. XSL, ca şi XML, se bazează pe etichete, deci pentru folosirea XSL sunt necesare aceleaşi abilităţi ca şi la crearea documentelor XML, accelerând astfel ritmul învăţării. Inima XSL este transformarea şi prelucrarea documentelor pentru medii diferite, deci XSL este capabil să gestioneze sisteme complexe de documente multimedia.

Sugestie de la LeeAnne.com Words to weave by

Complexitatea operaţiei este criteriul cel mai important la alegerea între CSS şi XSL. Dacă doriţi să realizaţi modificări vaste asupra documentului înainte de a-l prezenta, atunci XSL este instrumentul de care aveţi nevoie. Dar dacă doriţi să adăugaţi puţină strălucire unei pagini, lăsând structura de bază neatinsă, atunci CSS reprezintă probabil cea mai bună alegere.

Pe scurt, XSL este proiectat să formateze şi să transforme documentele XML în *instanţe* particulare ale unui document de bază. Specificaţia XSL completă este împărţită în două părţi principale: XSL propriu-zis şi XSLT. XPath, XML Path Language, este o parte integrantă a specificaţiei complete, dar din punct de vedere formal externă ei, deoarece acest modul este partajat de multe alte standarde. XPath a fost tratat în Capitolul 8, „XPath".

Conceptele fundamentale le veţi învăţa în acest capitol. Aplicaţiile CSS şi XSL le veţi cerceta în continuare în capitolele 13, „Foi de stil în cascadă şi XML/XHTML", respectiv 14, „Utilizarea generaţiei următoare de foi de stil extensibile – XSL".

Foi de stil în cascadă – CSS

Probabil că v-aţi familiarizat până acum cu CSS1 şi poate aţi utilizat şi CSS2. Acest capitol nu este o introducere în CSS şi nici un tutorial (îndrumar) care să acopere complet acest limbaj. El se concentrează pe un subiect restrâns: modul în care poate fi utilizat CSS pentru a formata documente XML. Multe browsere nu suportă cod XML, deci există numai câteva opţiuni.

Codul XML din Listingul 12.1 descrie înregistrarea de date utilizată în Capitolul 7, „Scheme XML", pentru a eticheta o înregistrare de client:

Listingul 12.1 O înregistrare de client în XML

```
<?xml version="1.0"?>
<?xml-stylesheet href="11xmp01.css" type="text/css" ?>
<!-- Customer Record -->
  <customer xmlns="http://www.leeanne.com/aristotelian/schemas/">
    <customer-name>Aristotelian Logic Systems</customer-name>
    <customerID>112344-0897</customerID>
    <credit-status>Net-30</credit-status>
    <customer-contact locale="USA">
      <honorific>Ms.</honorific>
      <given-name1>George</given-name1>
      <given-name2>Evelyn</given-name2>
      <family-name>Fayne</family-name>
```

```
    <degrees>LLD, PhD</degrees>
    <title>Contracts Director</title>
    <department>Purchasing</department>
    <telephone locale="NANP">=1 101 555.1234</telephone>
    <extension>3456</extension>
  </customer-contact>
  <comment>ALS buys over $300K per year, primarily our Mark IV WKS</comment>
  <shipping-address locale="USA">
    <attention>Attn:</attention>
    <attention-name>Receiving Clerk</attention-name>
    <reference>Ref:</reference>
    <street-address1>987 Technology Pkwy</street-address1>
    <street-address2>Unit 200 - Receiving</street-address2>
    <city>Riverdale</city>
    <territorial-division>Ohio</territorial-division>
    <territorial-abbreviation>OH</territorial-abbreviation>
    <postal-code>31089</postal-code>
    <country>USA</country>
  </shipping-address>
  <billing-address locale="USA">
    <attention>Attn:</attention>
    <attention-name>Accounts Payable</attention-name>
    <reference>Ref:</reference>
    <street-address1>987 Technology Pkwy</street-address1>
    <street-address2>Unit 300 - Accounting</street-address2>
    <city>Riverdale</city>
    <territorial-division>Ohio</territorial-division>
    <territorial-abbreviation>OH</territorial-abbreviation>
    <postal-code>31089</postal-code>
    <country>USA</country>
  </billing-address>
  <creation-date>19950711T034233-8</creation-date>
  <last-updated>19991011T112542-8</last-updated>
</customer>
```

Amintindu-ne că numele elementelor pot fi orice nume de etichete XML, putem utiliza o foaie de stil CSS1 pentru a realiza formatări de bază ale acestor date, așa cum se arată în Listingul 12.2.

Listingul 12.2 Un exemplu de foaie de stil CSS1 pentru înregistrarea de client XML

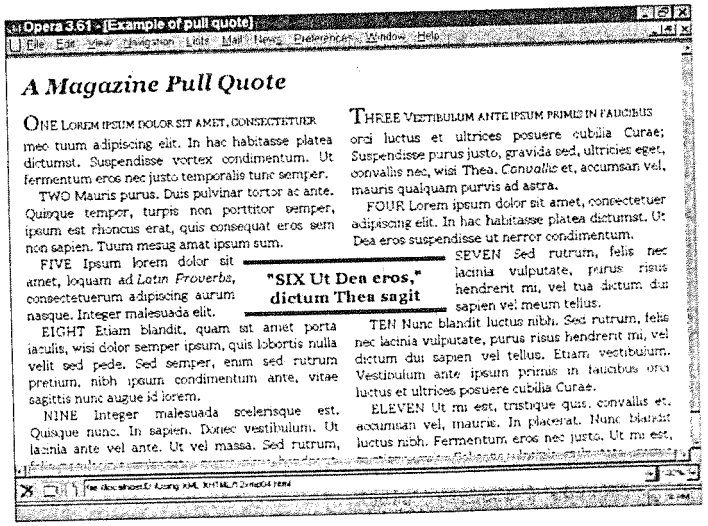
```
// Customer Record
  customer { float: left; clear: left; margin-left: 1pc; color: #000000; }
    customer-name { float: left; clear: left; }
    customerID { float: left; clear: left; margin-left: 1pc; }
    credit-status { float: left; clear: left; margin-left: 1pc; }
    customer-contact { float: left; clear: left; margin-left: 2pc; background:
➥#ffdddf; }
      honorific { float: left; clear: left; }
      given-name1 { float: left; clear: left; }
      given-name2 { float: left; clear: left; }
      family-name { float: left; clear: left; }
      degrees { float: left; clear: left; }
      title { float: left; clear: left; }
      department { float: left; clear: left; }
```



```
      telephone { float: left; clear: left; }
      extension { float: left; clear: left; }
      comment { float: left; clear: left; margin-left: 1pc; }
    shipping-address { float: left; clear: left; margin-left: 2pc; background:
➥#ddffff; }
      attention { float: left; clear: left; }
      attention-name { float: left; clear: left; }
      reference { float: left; clear: left; }
      street-address1 { float: left; clear: left; }
      street-address2 { float: left; clear: left; }
      city { float: left; clear: left; }
      territorial-division { float: left; clear: left; }
      territorial-abbreviation { float: left; clear: left; }
      postal-code { float: left; clear: left; }
      country { float: left; clear: left; }
    billing-address { float: left; clear: left; margin-left: 2pc; background:
➥#ffffdd; }
      attention { float: left; clear: left; }
      attention-name { float: left; clear: left; }
      reference { float: left; clear: left; }
      street-address1 { float: left; clear: left; }
      street-address2 { float: left; clear: left; }
      city { float: left; clear: left; }
      territorial-division { float: left; clear: left; }
      territorial-abbreviation { float: left; clear: left; }
      postal-code { float: left; clear: left; }
      country { float: left; clear: left; }
    creation-date { float: left; clear: left; margin-left: 1pc; }
    last-updated { float: left; clear: left; margin-left: 1pc; }
```

Figura 12.1 prezintă browserul Opera afișând înregistrarea de client formatată. Desigur, putem face formătări mult mai extinse asupra ieșirii, dar aceasta este suficientă pentru a vă forma o idee.

FIGURA 12.1
Browserul Opera afișează un document XML utilizând proprietățile CSS2 suportate.



Deoarece multe browsere nu implementează toate caracteristicile CSS1, nu vă puteți baza complet pe proprietatea display care să vă permită să impuneți ca un element să fie considerat un bloc, lucru pe care îl dorim de fapt. Totuși, acest

lucru poate fi simulat prin setarea proprietăților `float` și `clear`, care sunt suportate mai bine, la fiecare articol din listă. Rețineți că este doar una dintre numeroasele prezentări alternative ale datelor. Prin schimbarea foii de stil puteți formata același document în mai multe moduri și, mai mult, foile de stil permit utilizatorilor să înlocuiască foaia dumneavoastră de stil cu o foaie de stil personalizată, mult mai potrivită nevoilor lor individuale. Browserul Opera de la Opera Software din Norvegia realizează acest lucru foarte comod, prin butonul Swap Style Sheets (schimbă foile de stil) de pe bara de locații Web. Studiați-l la `http://www.operasoftware.com/`.

Sugestie de la LeeAnne.com Words to weave by

Având așa de multe alternative, alegerea caracteristicii pe care să o utilizați poate fi o problemă. O listă a caracteristicilor CSS care sunt relativ sigure, precum și a celor care sunt periculoase, apare pe situl Web Review's Style Sheets, la `http://webreview.com/pub/Style_Sheets`.

În CSS2 aveți mult mai mult control, deși în momentul de față nici un browser nu suportă funcționalitatea completă a CSS2. Utilizând CSS2 puteți realiza o transformare primitivă a textului, prin specificarea unui text care va fi inserat înainte sau după un element, utilizând o pseudo-clasă. Foaia de stil prezentată în Listingul 12.3 îmbunătățește versiunea CSS1 prin adăugarea altor comenzi de formatare, mai ales a posibilității de a eticheta câmpurile. Totuși, fiți atent, deoarece suportul oricărei caracteristici CSS2 este neuniform în browserele cele mai importante.

Notă

Chiar dacă browserele recente suportă o caracteristică, aceasta nu înseamnă prea mult dacă nu cunoașteți foarte bine publicul dumneavoastră. La momentul scrierii acestei cărți, numărul utilizatorilor browserelor Microsoft versiunea 4 sau una anterioară era aproape dublu față de cel al utilizatorilor de MSIE versiunea 5. Nu vă puteți baza pe modernizări, chiar dacă acestea sunt gratuite. Pentru multe persoane, e prea multă bătaie de cap.

Listingul 12.3 Un exemplu de foaie de stil CSS2 pentru înregistrarea de client XML

```
customer { display: block; margin-left: 1pc; margin-right: 3pc; color: #000000; }
    customer-name:before { content: "Customer Name: "; font-weight: bold; }
    customer-name { display: block; }
    customerID:before { content: "Customer ID: "; }
    customerID { display: block; margin-left: 1pc; }
    credit-status:before { content: "Credit Status: "; }
    credit-status { display: block; margin-left: 1pc; }
    customer-contact:before { content: "Customer Contact: ";  font-weight: bold; }
    customer-contact { display: block; margin-left: 2pc; background: #ffddff; }
      honorific:before { content: "Honorific: "; }
      honorific { display: block; }
      given-name1:before { content: "Given Name 1: "; }
      given-name1 { display: block; }
      given-name2:before { content: "Given Name 2: "; }
      given-name2 { display: block; }
      family-name:before { content: "Family Name: "; }
      family-name { display: block; }
      degrees:before { content: "Degrees: "; }
      degrees { display: block; }
```



```
    title:before { content: "Title: "; }
    title { display: block; }
    department:before { content: "Department: "; }
    department { display: block; }
    telephone:before { content: "Telephone: "; }
    telephone { display: block; }
    extension:before { content: "Extension: "; }
    extension { display: block; }
    comment:before { content: "Comment: "; }
    comment { display: block; margin-left: 1pc; }
  shipping-address:before { content: "Shipping Address: ";  font-weight: bold; }
  shipping-address { display: block; margin-left: 2pc; background: #ddffff; }
    attention:before { content: "Attention: "; }
    attention { display: block; }
    attention-name:before { content: "Attention Name: "; }
    attention-name { display: block; }
    reference:before { content: "Reference: "; }
    reference { display: block; }
    street-address1:before { content: "Street Address 1: "; }
    street-address1 { display: block; }
    street-address2:before { content: "Street Address 2: "; }
    street-address2 { display: block; }
    city:before { content: "City: "; }
    city { display: block; }
    territorial-division:before { content: "Territorial Division: "; }
    territorial-division { display: block; }
    territorial-abbreviation:before { content: "Territorial Abbreviation: "; }
    territorial-abbreviation { display: block; }
    postal-code:before { content: "Postal Code: "; }
    postal-code { display: block; }
    country:before { content: "Country: "; }
    country { display: block; }
  billing-address:before { content: "Billing Address: ";  font-weight: bold; }
  billing-address { display: block; margin-left: 2pc; background: #ffffdd; }
    attention:before { content: "Attention: "; }
    attention { display: block; }
    attention-name:before { content: "Attention Name: "; }
    attention-name { display: block; }
    reference:before { content: "Reference: "; }
    reference { display: block; }
    street-address1:before { content: "Street Address 1: "; }
    street-address1 { display: block; }
    street-address2:before { content: "Street Address 2: "; }
    street-address2 { display: block; }
    city:before { content: "City: "; }
    city { display: block; }
    territorial-division:before { content: "Territorial Division: "; }
    territorial-division { display: block; }
    territorial-abbreviation:before { content: "Territorial Abbreviation: "; }
    territorial-abbreviation { display: block; }
    postal-code:before { content: "Postal Code: "; }
    postal-code { display: block; }
    country:before { content: "Country: "; }
    country { display: block; }
```

Listingul 12.3 (continuare)

```
creation-date:before { content: "Creation Date: "; }
creation-date { display: block; margin-left: 1pc; }
last-updated:before { content: "Last Updated: "; }
last-updated { display: block; margin-left: 1pc; }
```

Figura 12.2 prezintă modul în care afectează această foaie de stil afișarea în Mozilla, platforma browser open-source creată de Netscape și de comunitatea de utilizatori. Căutați noutăți despre Mozilla pe situl lor Web, la `http://www.mozilla.org/`.

Notă Browserul Mozilla, actualmente într-un stadiu de dezvoltare pre-beta, nu este recomandat pentru cei slabi de inimă. Este lent dar destul de stabil, se blochează din când în când și se află încă în faza de dezvoltare. Oricum, este un concept foarte interesant. Rămâne de văzut dacă mișcarea open-software poate sau nu să susțină o sarcină atât de mare. Să le dorim noroc.

Browserul este actualizat cam la fiecare două sau trei săptămâni, deci din punctul de vedere al testării browserului Mozilla, este greu să păsești de două ori în același râu. Oricum, îmbunătățirile apar constant și se apropie data efectivă a lansării, când, conform planurilor, Mozilla va deveni în sfârșit Netscape 5.0.

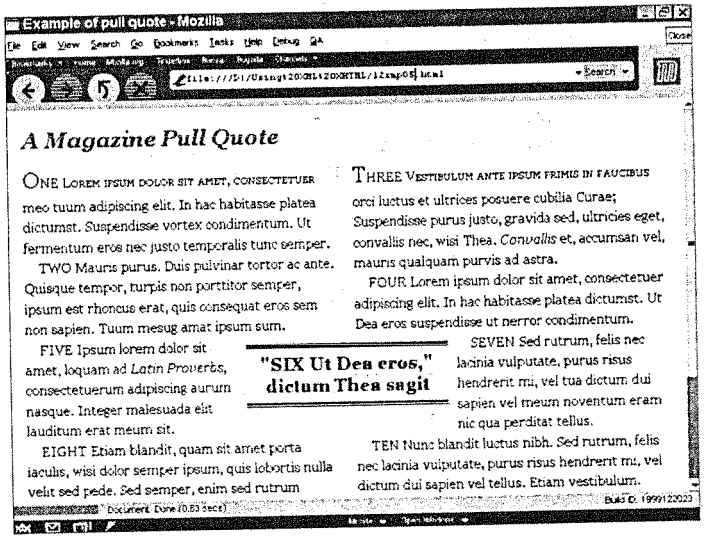


FIGURA 12.2
Ecranul Mozilla prezintă suportul său pentru proprietățile CSS2.

Dar nici acest lucru nu este suficient. Acoperirea lui CSS la orice nivel este atât de inconsistentă încât încercarea de a coda o foaie de stil complexă este un exercițiu de frustrare și probleme de afișare ascunse care apar numai după testări exhaustive. Bug-uri subtile de subordonare în implementările browserelor pot face ca problemele să apară pe pagină numai după câteva luni de afișări corecte, atunci când un articol fără legătură este adăugat sau modificat.

Limbajul extensibil al foilor de stil – XSL

Comunitatea de design Web a fost de acord că era nevoie de ceva care să suporte operațiile de stil complexe – acel ceva era XSL, limbajul extensibil al foilor de stil. *Complexe* înseamnă documente care necesită prelucrări vaste înainte de a putea fi afișate, cum ar fi

un ziar sau o revistă cu articole care se continuă pe paginile următoare. Înseamnă, de asemenea, și documente în limbi (cum este chineza) care nu respectă neapărat convenția de direcție orizontală a textului, utilizată de majoritatea scrierilor alfabetice. Era nevoie de trucuri de afișare cum ar fi direcția rândurilor de sus în jos (sau de jos în sus), metode de construire a cuvintelor din caractere neliniare (ca în coreeană) și altele. XSL este rezultatul.

Ca un exemplu, XSL și XSLT pot fi utilizate pentru a genera automat un cuprins și un index pe baza unui document XML. Obiectele de formatare implică faptul că, acum, mai multe pagini pot fi formatate pe loc, pornind de la un singur document de bază. Coloanele și casetele editoriale pot sta separat de conținut, ca în ziarele moderne sau în reviste, iar conținutul să se deruleze în „goluri" pre-poziționate. De exemplu, un articol de pe pagina 1 poate continua pe pagina 10. Stilul de ziar chinezesc, cu formatări de linie de sus în jos, poate fi manevrat de browserul utilizatorului, în loc să fie codat hard în mod laborios și plasat manual. Deoarece conceptele stânga, dreapta, sus și jos sunt asociate foarte strâns cu direcția de scriere, XSL a extins aceste concepte cu înainte, după, început și sfârșit. Ultimele concepte sunt neutre din punctul de vedere al limbii, deci o foaie de stil poate fi utilizată (teoretic) pentru a genera atât texte în stil englezesc, cât și în stil chinezesc.

Sugestie de la LeeAnne.com Words to weave by — Această flexibilitate de stil este un alt motiv pentru a alege XSL măcar pentru o etapă a procesului de creare a documentului dumneavoastră. Nici o altă tehnologie suportată și larg răspândită nu manevrează tipărirea de sus în jos și nici nu combină caracterele separate ale alfabetului coreean în blocuri suprapuse care constituie cuvinte.

Dar mai există ceva. În loc să învățați un limbaj complet nou pentru a stabili un stil pentru o pagină, XSL seamănă foarte mult cu familiarele etichete XML și HTML. Ca și XML, XSL are un DTD, deci poate fi validat, iar un browser care pretinde că este conform cu specificația poate fi testat în privința unei compatibilități reale. Pe de altă parte, asemenea altor standarde legate de XML, XSL se bazează pe structura documentului sau documentelor de bază. Utilizarea XSL și XSLT pentru a conferi un stil documentelor care nu sunt construite corect poate duce la rezultate neașteptate.

Sugestie de la LeeAnne.com Words to weave by — XSL încorporează aproape totul din CSS1 și CSS2 și adaugă caracteristici semnificative și din CSS3. Utilizând XSL, puteți realiza aproape orice operație de formatare care poate fi realizată cu CSS și chiar mai mult. XSLT, limbajul de transformare înrudit, vă permite să transformați literalmente documentul XML în ceva complet diferit. Este utilizat deseori pentru a „traduce" XML în HTML, pentru a fi afișat utilizând browsere moștenite.

Codul următor ilustrează simplitatea (și limbuția relativă) utilizării XSL pentru a specifica secțiunile de document care să primească prelucrări de stil în textul XML al *Comediei erorilor* de Shakespeare, plasat în domeniul public de Jon Bosak.

Iată o secțiune scurtă utilizată în acest exemplu fără rădăcina și fără porțiunile inferioare ale documentului XML.

Listingul 12.3A Un fragment XML din *Comedia erorilor* de Shakespeare

```
<?xml version="1.0"?>
<!DOCTYPE PLAY SYSTEM "play.dtd">
<PLAY>
<TITLE>The Comedy of Errors</TITLE>
<SCENE>
<TITLE>SCENE I.  The house of ANTIPHOLUS of Ephesus.</TITLE>
<STAGEDIR>Enter ADRIANA and LUCIANA</STAGEDIR>

<SPEECH>
<SPEAKER>ADRIANA</SPEAKER>
<LINE>Neither my husband nor the slave return'd,</LINE>
<LINE>That in such haste I sent to seek his master!</LINE>
<LINE>Sure, Luciana, it is two o'clock.</LINE>
</SPEECH>

<SPEECH>
<SPEAKER>LUCIANA</SPEAKER>
<LINE>Perhaps some merchant hath invited him,</LINE>
<LINE>And from the mart he's somewhere gone to dinner.</LINE>
<LINE>Good sister, let us dine and never fret:</LINE>
<LINE>A man is master of his liberty:</LINE>
<LINE>Time is their master, and, when they see time,</LINE>
<LINE>They'll go or come: if so, be patient, sister.</LINE>
</SPEECH>

<SPEECH>
<SPEAKER>ADRIANA</SPEAKER>
<LINE>Why should their liberty than ours be more?</LINE>
</SPEECH>

<SPEECH>
<SPEAKER>LUCIANA</SPEAKER>
<LINE>Because their business still lies out o' door.</LINE>
</SPEECH>

<SPEECH>
<SPEAKER>ADRIANA</SPEAKER>
<LINE>Look, when I serve him so, he takes it ill.</LINE>
</SPEECH>
<SPEECH>
<SPEAKER>LUCIANA</SPEAKER>
<LINE>O, know he is the bridle of your will.</LINE>
</SPEECH>

<SPEECH>
<SPEAKER>ADRIANA</SPEAKER>
<LINE>There's none but asses will be bridled so.</LINE>
</SPEECH>
</SCENE>
</PLAY>
```


Iar aici este foaia de stil XSL efectivă care afișează ieșirea prezentată în Figura 12.3.

Listingul 12.3B O foaie de stil XSL pentru piesa de teatru

```
<xsl:stylesheet
    xmlns:xsl='http://www.w3.org/XSL/Transform/1.0'
    xmlns:fo='http://www.w3.org/XSL/Format/1.0'
    result-ns="fo">

  <xsl:template match='/'>
    <fo:display-sequence
                start-indent='6pt'
                end-indent='6pt'
                font-size='12pt'>
      <xsl:apply-templates/>
    </fo:display-sequence>
  </xsl:template>

  <xsl:template match="TITLE">
    <fo:block   font-size="18pt"
                background-color="#ffff00"
                start-indent="24pt">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match="LINE">
    <fo:block   start-indent="32pt">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match="SPEAKER">
    <fo:block   space-before="12pt">
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

</xsl:stylesheet>
```

Vă rog să observați că nici unul dintre browserele importante nu suportă actual obiectele de formatare, care sunt necesare pentru a vizualiza aceste stiluri XSL. Utilizați puternicul browser InDelv de la InDelv, la `http://www.indelv.com/` (acum în versiune beta, dar va fi scos pe piață în curând) pentru a vedea genurile de formatări care pot fi realizate, după cum se vede și în Figura 12.3.

Notă Editorul/browserul XML/XSL InDelv este unul dintre cele mai bine proiectate și mai ușor de utilizat editoare/browsere XML, aflat încă în versiune beta. Puteți comuta între modurile de vizualizare a documentului pentru a cerceta foaia de stil XSL, pentru a studia codul sursă și chiar pentru a importa pagini Web aleatoare cu o curățare automată și raportarea erorilor. Este destul de impresionant chiar și în această formă intermediară. Îl puteți găsi la `http://www.indelv.com`.

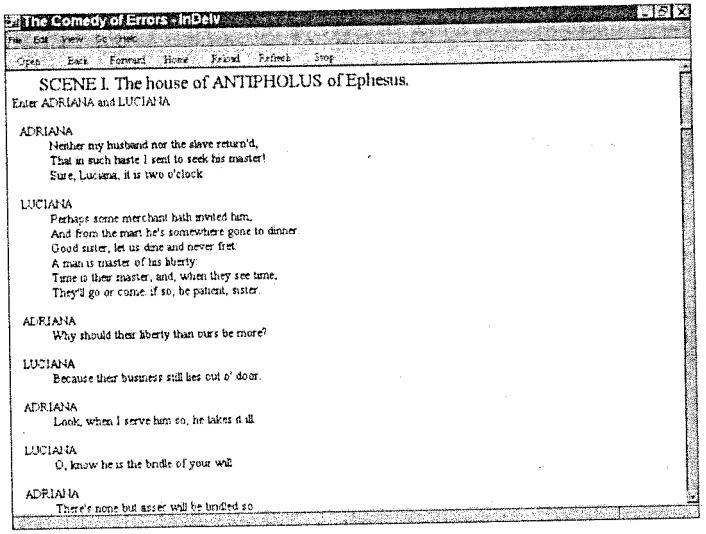


FIGURA 12.3
Browserul InDelv prezentând o pagină formatată cu XSL.

Versiunea de producție a editorului InDelv era planificată să fie lansată în vara lui 2000, deci ar trebui să fie disponibilă în formă completă în momentul în care citiți aceste rânduri. Versiunea intermediară a browserului InDelv funcționează și este disponibilă pentru descărcare pe situl lor.

Limbajul de semantică și specificare a stilului documentului – DSSSL (Document Style Semantics and Specification Language)

DSSSL este un strămoș al XSL și este încă utilizat de mulți specialiști în documente. Din DSSSL provin obiectele de formatare utilizate de XSL, deși în DSSSL sunt numite obiecte flux. DSSSL este extraordinar de complex, necesitând mai multe deprinderi de programare decât XSL, dar, ca urmare, este și mai puternic. Multe comunități de utilizatori utilizează DSSSL în mod curent pentru a realiza multe transformări ale documentului, necesare în anumite etape ale procesului de publicare, de exemplu, sau pentru a transforma date în formă canonică pentru stocare sau transfer.

Sugestie de la LeeAnne.com Words to weave by

Ca și în cazul SGML, dacă organizația dumneavoastră are nevoie să utilizeze DSSSL, probabil că deja îl utilizați. Pentru marea majoritate, DSSSL este mult prea ezoteric pentru a fi utilizat cu ușurință. XSL este un compromis bun între putere și ușurință în utilizare, utilitatea fiind cel mai important considerent pentru multe organizații.

Alegerea unui limbaj de stil

Alegerea între CSS, XSL, DSSSL și limbajele de scriptare poate fi constrânsă de mediul dumneavoastră sau de necesitățile proiectului dumneavoastră. Fiecare mecanism de stil are atât puncte tari, cât și puncte slabe. Puteți să vă gândiți la procese formate din doi sau trei pași, realizarea unui format de stil inițial cu XSL, apoi utilizarea lui XSLT pentru a genera o pagină XHTML având atașată o foaie de stil CSS pentru a asigura un suport larg pe durata tranziției de la capriciosul suport CSS la suportul larg al standardului XSL stabil.

Sugestie de la LeeAnne.com Words to weave by

Acest proces treptat este chiar cel utilizat de multe edituri importante. Un autor predă un document marcat cu un set simplu de etichete, apoi etichetele sunt îmbogățite de către un editor cu etichete mai complexe pentru un proces de editare internă, iar în final, etichetele îmbogățite și textul sunt transformate de un alt editor sau de un tehnoredactor într-un limbaj de tehnoredactare specializat, utilizat de tipografie.

Suport pentru limbaje de stil

Am mai vorbit în trecere despre acest lucru, dar oricât am insista, nu am reușit să spunem totul; suportul pentru limbajele de stil nu este universal. Versiunea actuală a browserului Microsoft Internet Explorer utilizează un model XSL depășit, care este echivalentul unui limbaj propriu. Deși poate fi recunoscut ca fiind XSL, este totuși suficient de diferit pentru ca anumite caracteristici să nu funcționeze sau să funcționeze altfel decât s-ar dori. Netscape nu oferă nici un fel de suport, decât în varianta open-source Mozilla, vag înrudită cu acesta. Opera nu va include suport XSL decât după ce schița de lucru W3C va deveni o recomandare.

Anterior, Figura 12.2 a arătat că Mozilla are încorporat un suport destul de bun pentru XML și CSS2, dar paginile nu pot fi afișate corect în nici o versiune actuală de Internet Explorer, Netscape sau Opera. Paragrafele următoare aruncă o privire asupra unei operații CSS1 simple și asupra modului în care aceasta este tratată de Netscape și de Microsoft, precum și de Opera, deoarece acest browser este frecvent utilizat atât de persoane cu handicap de vedere, cât și pe plan internațional.

Exemplul este o pagină foarte simplă care are un element mobil. El există în lumea reală, deci veți vedea compromisurile care au fost făcute pentru ca totul să funcționeze fără a fi nevoie de tabele. După cum am mai spus, în tabele se navighează dificil și sunt înțelese cu greu de către persoanele cu handicap de vedere, deci ar trebui evitată pe cât posibil utilizarea lor ca elemente de machetă. Din păcate, implementarea neatentă a elementelor mobile de către mulți producători de browsere a făcut ca tabelele să fie mult mai ușor de utilizat decât elementele mobile, deși elementele mobile ar fi o metodă mult mai bună dacă ar fi disponibilă. Elementele `div` sunt utilizate pentru a dubla efectul unui element container extern ca și când ar fi un alt tip de fișier XML, dar este totuși XHTML, deci orice browser îl poate reda într-o *anumită* măsură.

Listingul 12.4 prezintă foaia de stil și codul pentru o pagină simplă care utilizează CSS.

Listingul 12.4 Un exemplu de pagină Web care utilizează XHTML și CSS in-line

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/strict.dtd">
<html
 xmlns="http://www.w3.org/TR/xhtml1">
  <head>
    <title>
      Tabor Sarah Books
    </title>
<style
 type="text/css">
     <!--
      body { font-family: serif; font-size: 18pt; margin: 10pt; }
```

Listingul 12.4 (continuare)

```
      h1 { font-size: 32pt; }
      img { float: left; margin-right: 10pt; }
      div.ritual { margin: 10pt; }
      div.origin { margin-left: 20pt; }
      div.publisher { margin-left: 20pt; }
      div.order { margin: 20pt; }
      div.address { margin: 20pt; }
      div.price { margin-left: 20pt; }
      div.title { margin: 10pt; clear: left;  }
      div.telephone { margin: 10pt; }
      div.fineprint { margin: 10pt; font-size: 10pt; }
      div.linkout { margin: 20pt; }
    // -->
</style>
</head>
  <body>
    <h1>Tabor Sarah Books</h1>
    <img src="splash.gif" height="365" width="54"
     alt="[Image: A stream of water splashes into an open hand]" />
    <div class="ritual">
      <a href="ritual.html"><cite>A Ritual of Drowning - Poems of
      Love and Mourning</cite></a><br />
      <div class="origin">
        by Teya Schaffer<br />
        <div class="publisher">
          Tabor Sarah Books, 1999, Oakland and Palo Alto<br />
          ISBN: Ø-935079-18-1<br />
        </div>
      </div>
      <div class="order">
        This book can be ordered from:
        <div class="address">
          Tabor Sarah Books<br />
          367 50th Street<br />
          Oakland CA 94609<br />
        </div>
        <div class="price">
          The current price is $10.95 US domestic postage paid.
        </div>
      </div>
    </div>
    <div class="title">
      <cite>Chag Sameach! a Jewish holiday book for
      children</cite><br />
      <div
       class="origin">
        by Patricia Schaffer<br />
        <div class="publisher">
          Tabor Sarah Books, 1985, Oakland and Palo Alto<br />
        </div>
      </div>
      <div class="order">
```



```
        This book can be ordered from:
        <div
         class="address">
          Tabor Sarah Books<br />
           3345 Stockton Place<br />
           Palo Alto, CA 94303
        </div>
        <div class="price">
          The current price is $$7.70 US domestic postage paid.
        </div>
      </div>
    </div>
    <div class="title">
      <cite>How Babies and Families Are Made - There is more than
      one way!</cite><br />
      <div class="origin">
        by Patricia Schaffer<br />
         illustrated by Suzanne Corbett<br />
        <div class="publisher">
          Tabor Sarah Books, 1988, Oakland and Palo Alto
        </div>
      </div>
      <div class="order">
        This book can be ordered from:
        <div class="address">
          Tabor Sarah Books<br />
           3345 Stockton Place<br />
           Palo Alto, CA 94303
        </div>
        <div class="price">
          The current price is $$9.70 US domestic postage paid.
        </div>
      </div>
    </div>
    <div class="telephone">
      Phone inquiries for all titles: +1 650 494-7846<br />
       Titles are also available through women's bookstores.
    </div>
    <div class="fineprint">
      <hr />
      Page Copyright &#169; 1999, by Lee Anne Phillips
      <a href=
      "mailto:Lee%20Anne%20Phillips%20&lt;leeanne@leeanne.com&gt;">
      &lt;leeanne@leeanne.com&gt;</a> - All Rights Reserved
      Worldwide
      <p>
         Illustration is from <cite>A Ritual of Drowning - Poems of
        Love and Mourning</cite> Copyright &#169; 1999 by Teya
        Schaffer - All Rights Reserved Worldwide
      </p>
    </div>
    <div class="linkout">
      <p>
         For more reviews of outstanding books by women, please
         visit the <a
```

Listingul 12.4 (continuare)

```
        href="http://www.cybergrrl.com/review/">Women's Books
        Online</a> pages kindly hosted by <a
        href="http://www.cybergrrl.com/">Cybergrrl!</a>
      </p>
    </div>
  </body>
</html>
```

În Figura 12.4, prezentăm aspectul paginii în browserul Netscape actual. Arată destul de rău. Acest cod poate fi găsit pe Web, la `http://www.leeanne.com/usingxml/index1.html`, pentru a vedea modul în care este tratat de browserul dumneavoastră actual.

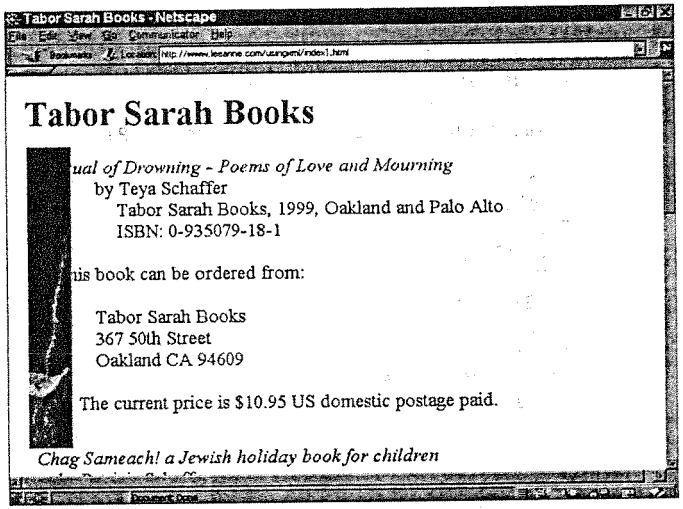


FIGURA 12.4
Netscape furnizează un afişaj CSS cu probleme serioase de lizibilitate.

Priviți versiunea MSIE din Figura 12.5. În ciuda faptului că este lizibilă, nu face totuși ceea ce ar trebui.

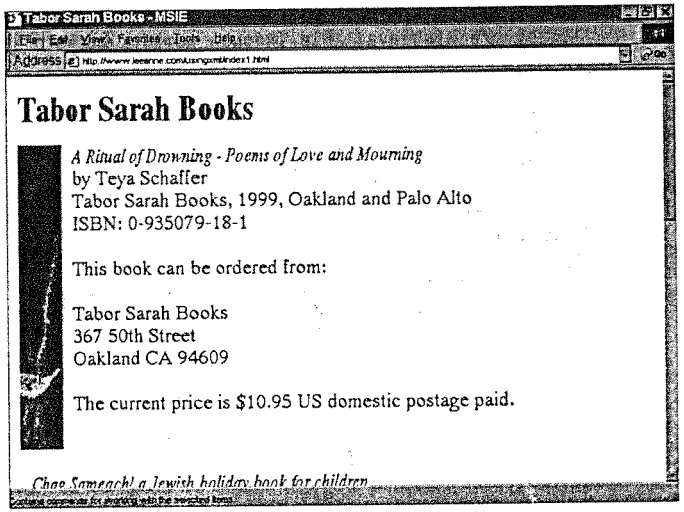


FIGURA 12.5
Microsoft Internet Explorer este mai lizibil, dar ignoră insertul atunci când prelucrează imaginea mobilă, o eroare de formatare destul de gravă. Deşi este mai puțin deranjant, se îndreaptă totuşi spre un afişaj urât.

Figura 12.6 prezintă pagina în versiunea actuală a browserului Opera. Este foarte asemănătoare cu versiunea Microsoft IE, excepție făcând faptul că strică linia de text, care trebuia să aibă o margine de 10 puncte, aproape la fel de rău cum face Netscape cu liniile precedente.

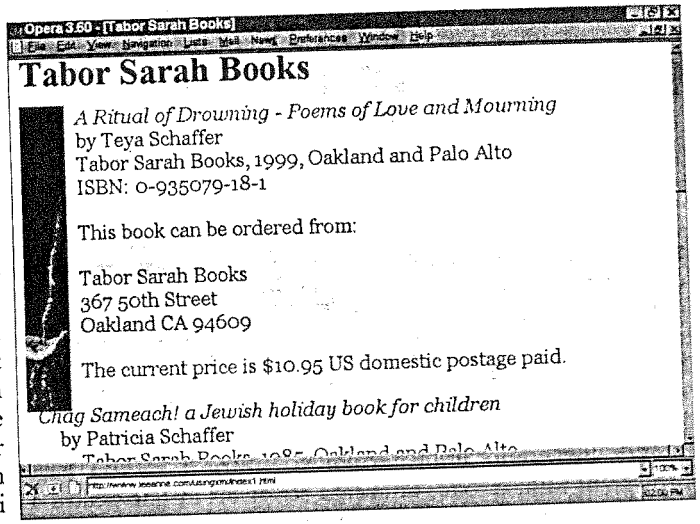


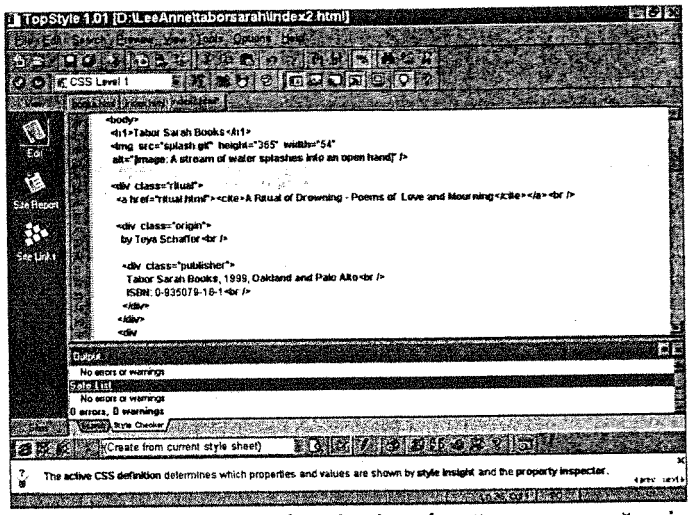
FIGURA 12.6
Opera afişează pagina cu erori de formatare şi cu probleme de lizibilitate minore.

Nici una dintre aceste implementări nu este curată şi fiecare strică afişarea într-un mod sau altul, deşi toate sunt produse „mature". În exemplul ales, Netscape merge cel mai rău, dar este uşor să alegeți un alt set de caracteristici care să lase în dezavantaj celelalte browsere. Toate cele trei browsere manevrează marginile într-un mod nepotrivit în apropierea unui element mobil.

În Figura 12.7 puteți vedea rezultatele verificării foii de stil cu TopStyle, un instrument proiectat să identifice problemele foilor de stil şi chiar să realizeze câteva verificări pentru mai multe browsere. Din păcate, foaia de stil trece fără erori, deci este ceva ce TopStyle nu poate depista. De vină nu este instrumentul, care este foarte valoros, ci complexitatea testării fiecărei combinații posibile de proprietăți, în fiecare mod posibil. În cod există 13 proprietăți dintr-un total de 55 de opțiuni disponibile, fiecare având în medie 10 valori. Asta înseamnă 9,4×10 la puterea 34, sau, mai simplu, aproximativ 90.365.466.692.518.617.600.000.000.000.000.000 de combinații, plus minus câteva catralioane. Ar trebui efectuate teribil de multe teste.

În ciuda faptului că aceste probleme pot fi ajustate până când vor funcționa corect, codul rezultat este dezordonat. De asemenea, trebuie să utilizați script-uri pentru a implementa coduri specifice browserelor sau să evitați complet utilizarea marginilor, înlocuindu-le cu marcaje de bloc, aceasta fiind o utilizare improprie, dar foarte obișnuită a acestei caracteristici XHTML. Cu implementările actuale ale foilor de stil în cascadă, aveți deseori probleme, indiferent de ceea ce faceți. Aproape fiecare caracteristică CSS are probleme similare atunci când vă gândiți la suportul pentru mai multe browsere. În Anexa D, „Referințe CSS1, CSS2", perechile proprietate şi valoare care sunt relativ sigure în utilizare sunt scrise cu aldine. Veți observa că numărul de perechi scrise cu aldine nu este copleşitor.

FIGURA 12.7
TopStyle raportează probleme la multe foi de stil, dar nu găsește nici una în această foaie de stil. Mă rog!



Codul care funcționează este ajustat pentru a reduce problemele compatibilității cu mai multe browsere la un nivel acceptabil și poate fi văzut pe Web, la `http://www.leeanne.com/usingxml/11xmp04.html`. Multe situri reale găsesc necesară utilizarea unui instrument special server-side de rutare, cum este BrowserHawk, pentru a depista browserul utilizatorului și pentru a furniza pagini diferite în funcție de browserul utilizat. Sau, utilizează script-uri in-line pentru a alege între coduri in-line diferite, bazate pe browser. În termeni moderați, este o bătaie de cap, dar măcar funcționează. Într-un fel.

Sugestie de la LeeAnne.com Words to weave by

W3C are disponibil un set de instrumente de testare CSS pe situl său, la `http://jigsaw.w3.org/css-validator/` și la `http://www.w3.org/Style/css/Test/`. Puteți utiliza, de asemenea, un instrument de validare comercial cum este TopStyle, de la `http://www.bradsoft.com/topstyle/`.

Pentru ultimul exemplu, foaia de stil CSS2 este translatată direct în XSL, după cum se vede în Listingul 12.5. Deoarece XSL implementează mai mult sau mai puțin direct majoritatea caracteristicilor CSS2, de obicei este ușor să translatați un limbaj la altul după ce cunoașteți sintaxa corespunzătoare. Observați, din nou, că XSL a extins conceptele de margine stânga, dreapta, sus și jos, cu înainte, după, început și sfârșit, pentru a reduce neplăcutele supoziții dependente de limbă (deși modelul mai vechi este încă disponibil). Proprietățile marginilor păstrate pentru compatibilitatea cu CSS sunt utilizate numai pentru a face foile de stil mai ușor de comparat. De asemenea, definițiile de stil sunt adăugate pentru câteva elemente XHTML deoarece nu vă puteți baza pe un browser XML adevărat pentru a le afișa într-un anumit mod.

Listingul 12.5 O foaie de stil XSL pentru un document XML

```
<?xml version='1.0'?>

<xsl:stylesheet
    xmlns:xsl='http://www.w3.org/XSL/Transform/1.0'
    xmlns:fo='http://www.w3.org/XSL/Format/1.0'
    version='1.0'>
```



```
<xsl:output method='fo'/>

  <xsl:template match='/'>
    <fo:display-sequence
        margin='10pt'
        font-family='serif'
        font-size='18pt'>
      <xsl:apply-templates/>
    </fo:display-sequence>
  </xsl:template>

  <xsl:template match="h1">
     <fo:display-sequence
        font-size='32pt'>
      <xsl:apply-templates/>
    </fo:display-sequence>
  </xsl:template>

  <xsl:template match="div[@class='ritual']">
    <fo:block
        margin='10pt'>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match="div[@class='origin']">
    <fo:block
        margin-left='20pt'>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match="div[@class='publisher']">
    <fo:block
        margin-left='20pt'>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

<xsl:template match="div[@class='order']">
     <fo:block
         margin='20pt'>
       <xsl:apply-templates/>
     </fo:block>
  </xsl:template>

    <xsl:template match="div[@class='address']">
      <fo:block
          margin='20pt'>
        <xsl:apply-templates/>
      </fo:block>
    </xsl:template>
```

Listingul 12.5 (continuare)

```
  <xsl:template match="div[@class='price']">
    <fo:block
        margin-left='20pt'>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match="div[@class='title']">
    <fo:block
        clear='left'
        margin='10pt'>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

<xsl:template match="div[@class='telephone']">
    <fo:block
        margin='10pt'>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match="div[@class='fineprint']">
    <fo:block
        margin='10pt'>
        font-size='10pt'>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match="div[@class='linkout']">
    <fo:block
        margin='20pt'>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match='p'>
    <fo:block
        space-end='6pt'
        space-before='6pt'
        space-after='6pt'>
      <xsl:apply-templates/>
    </fo:block>
  </xsl:template>

  <xsl:template match='cite'>
    <fo:inline-sequence font-style='italic'>
      <xsl:apply-templates/>
    </fo:inline-sequence>
  </xsl:template>
```



```
  <xsl:template match='hr'>
    <fo:display-rule
        margin='10pt'
        rule-thickness='1.5pt'>
      <xsl:apply-templates/>
    </fo:display-rule>
  </xsl:template>

  <xsl:template match='a'>
    <fo:simple-link
        external-destination='{@href}'
        color='rgb(0,0,255)'>
      <xsl:apply-templates/>
    </fo:simple-link>
  </xsl:template>

</xsl:stylesheet>
```

Codul XSL este mult mai bogat decât CSS, după cum puteți vedea cu ușurință, deoarece foaia de stil CSS2 echivalentă are nevoie numai de 13 linii!

Combinarea XHTML şi XML pentru prelucrări de stil

Deoarece deseori browserele mai vechi nu suportă deloc XML, iar XSL și mai puțin, aplicațiile care doresc să aibă cea mai mare audiență, păstrând însă câteva din avantajele XML, combină conținutul XML cu etichetele lor XHTML. Acest lucru permite etichetelor XHTML să manevreze cel mai scăzut nivel de formatare posibil, păstrând însă flexibilitatea XML în etichetarea datelor și în căutări. Evident, documentul trebuie să fie construit corect, dar acesta nu prea este un dezavantaj deoarece codul HTML fără erori tinde oricum să fie corect în conformitate cu DTD-ul HTML. În exemplele din ultima secțiune, în locul laborioaselor secțiuni `div` puteți eticheta datele dumneavoastră cu etichete XML, utilizând o zonă de nume XML, precum și cu etichetele de machetare XHTML de bază. Astfel puteți realiza cu ușurință asupra documentului căutări și transformări stil XSL, în timp ce permiteți ca documentul să fie vizualizat de milioane de browsere moștenite, utilizate încă pe Web. Vă rog să rețineți că printre acestea există și multe browsere utilizate de persoane cu necesități de accesibilitate speciale, deci nu puteți scăpa de ele punând cu obrăznicie un logo „vizualizat cel mai bine cu browserul X“ și o legătură pentru descărcare.

Sugestie de la LeeAnne.com Words to weave by

Așa cum nu toată lumea folosește același parfum sau aceeași apă de colonie, nu toată lumea utilizează același browser. Unii utilizează browsere bazate pe text sau auditive, uneori pentru că sunt nevoiți iar alteori pentru că acestea sunt la îndemână. Natura omniprezentă a PDA-urilor (asistent personal digital) navigatoare pe Web și posibila includere a browserelor auditive până și în telefonul celular sau în mașina dumneavoastră face imperativă luarea în considerare a tuturor utilizatorilor posibili atunci când proiectați propriile dumneavoastră pagini. Un Palm Pilot nu este un WebTV, iar un WebTV nu este un Power Mac G4.

Preluând această abordare, Listingul 12.6 prezintă acest fragment de cod cu etichete XML combinate cu sosiile lor XHTML.

Listingul 12.6 Un exemplu de document XHTML cu incluziuni XML și o foaie de stil in-line prestabilită

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="taborsarah.xsl"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    " http://www.w3.org/1999/xhtml ">
<html
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:tabor="http://www.leeanne.com/taborsarah/">
  <head>
    <title>Tabor Sarah Books</title>
<style
 type="text/css">
     <!--
      body { font-family: serif; font-size: 18pt; }
      h1 { font-size: 32pt; }
      img { float: left; }
      div.fineprint { font-size: 10pt; }
      div.origin { margin-left: 20px; }
      div.title { clear: left; font-size: 18pt; }
    // -->
</style>
    <meta http-equiv="PICS-Label"
       content='(PICS-1.1 "http://www.rsac.org/ratingsv01.html" l
    gen true r (n 0 s 0 v 0 l 0)' />
    <meta http-equiv="PICS-Label"
       content='(PICS-1.1 "http://www.classify.org/safesurf/" l gen
    true r (SS~~000 1))' />
  </head>
  <body>
    <blockquote>
      <tabor:page-header><h1>Tabor Sarah Books</h1></tabor:page-header>
      <tabor:image><img src="splash.gif" height="365" width="54"
       alt="[Image: A stream of water splashes into an open
            hand]" /><tabor:image>
      <div class="ritual">
        <blockquote>
          <blockquote>
            <tabor:link><a href="ritual.html"><cite>A Ritual of Drowning
               - Poems of Love and Mourning</cite></a></tabor:link><br />
            <blockquote>
              by <tabor:author>Teya Schaffer</tabor:author><br />
              <tabor:publisher>Tabor Sarah Books</tabor:publisher>,
              <tabor:publish-date>1999<tabor:publish-date>,
              <tabor:publish-city>Oakland and Palo Alto<tabor:publish-city>
              ISBN: <tabor:ISBN>0-935079-18-1</tabor:ISBN><br />
              <br />
            </blockquote>
            <blockquote>
```



```
              This book can be ordered from:
              <blockquote>
                <tabor:order-address>Tabor Sarah Books<br />
                367 50th Street<br />
                Oakland CA 94609<br /></tabor:order-address>
              </blockquote>
              The current price is <tabor:price>$10.95</tabor:price>
              US domestic postage paid.
            </blockquote>
          </blockquote>
        </blockquote>
      </div>
    </blockquote>
  </body>
</html>
```

Observați că mult text XHTML apare în afara etichetelor XML. Utilizând XSLT (sau CSS2) puteți genera aceste includeri de text străin și din foaia dumneavoastră de stil, deci nu prea aveți nevoie de ele pentru a reda documentul. Dacă aceste date le stocați într-o bază de date externă, atunci probabil că nu doriți să aveți informații redundante stocate în mai multe locuri.

Sugestie de la LeeAnne.com Words to weave by

Pentru informații repetitive, cum ar fi afișarea unui catalog, XSL poate reduce foarte mult cantitatea de cod, chiar dacă utilizatorii nu văd niciodată XML în mod direct. Cu aptitudinea XSL de a manipula și transforma documentele pe loc, el poate fi ideal pentru generarea mai multor pagini dintr-o bază de date utilizând un șablon partajat.

Deși poate va trece ceva timp până când adevăratele browsere XML vor fi larg utilizate, ele există sub forma browserelor SGML, care pot face cu ușurință trecerea la XML și, în multe cazuri, chiar au făcut-o. Având adăugat suportul pentru XSL în completarea suportului existent pentru DSSSL și pentru foile de stil brevetate, este posibil să vă faceți o idee despre cum va fi navigarea XML în viitorul apropiat. Mozilla suportă deja unele dintre standardele noi și promite să suporte mai multe o dată ce recomandările vor ieși din mâinile W3C, și există companii care lucrează la acest subiect chiar în acest moment. De exemplu, CiTEC (http://www.citec.fi/) din Finlanda are deja disponibil un browser XML/SGML, MultiDoc Pro, iar în viitorul apropiat va fi pus în circulație un produs bazat pe Doczilla.

O mare parte din entuziasmul și suportul pentru XML există în afara Statelor Unite, Canadei și Marii Britanii, deoarece utilizatorii din statele menționate sunt deja satisfăcuți de serviciile browserelor existente. Finlanda are legături puternice cu Rusia și cu alte părți ale fostei Uniuni Sovietice, deci suportul pentru scrierea chirilică și pentru mai multe limbi face XML mult mai atractiv decât ar putea fi pentru un utilizator oarecare dintr-o lume vorbitoare de limba engleză.

Utilizarea în tandem a CSS și XSL

Mai observați că exemplul anterior utilizează atât foi de stil CSS in-line, cât și o foaie de stil XSL externă. Acest lucru permite suport pentru multe browsere moștenite, în timp ce păstrează o funcționalitate maximă pentru utilizatorii cu browsere care au suport XML/XSL.

Așa cum s-a menționat anterior în mai multe locuri, există deja browsere care suportă XML, cum ar fi MultiDoc Pro sau InDelv. Mozilla și CiTEC lucrează împreună la Doczilla, care este versiunea sau modulul XML al produsului complet Mozilla. Deși InDelv este singurul browser XML adevărat care suportă XSL, probabil că celelalte vor recupera întârzierea până în momentul în care citiți aceste rânduri. Prea multe se bazează pe XML pentru ca acesta să rămână în urmă din cauza lipsei de suport în browsere, și multe companii se pregătesc să vă ia banii pentru cele mai grozave browsere cu suport XML.

Notă Există contradicții raționale și motivate referitoare la nivelul de suport XML la care vă puteți aștepta în viitorul imediat, acest text având o tentă optimistă. Unii sunt de părere că browserele complet compatibile XML vor penetra încet pe piața utilizatorilor, în parte din cauza competiției cu numeroasele browsere HTML gratuite, care și-au dovedit utilitatea de-a lungul timpului și sunt disponibile sau chiar instalate la momentul actual. Totuși, noi considerăm că avantajele suportului XML vor învinge această inerție.

Pe de altă parte, suportul inițial pentru XSL și XSLT poate fi localizat pe server și invizibil utilizatorilor, deoarece o soluție interimară pentru servirea de pagini XML utilizatorilor HTML este realizarea transformării pe server și furnizarea pe desktop a unui HTML curat. Desigur, prin acest lucru se renunță la aproape toată puterea lui XML, deci oamenii care plătesc fericiți sute de dolari pentru a avea la calculatorul lor boxe cu subwoofere și cu sunet surround este posibil să nu fie dispuși să amâne suportul XML complet de dragul câtorva dolari.

Concluzii

Toate limbajele de stil au același scop: permiterea separării conținutului de afișaj. XML/XSL extind acest scop principal cu o flexibilitate și o putere extraordinare. Ați văzut deja cum combinarea etichetelor XML vă permite să formatați cu ușurință diferite fragmente de cod în diferite moduri, chiar dacă acestea sunt în mod fundamental text. Dacă aveți Word 2000, atunci încercați să salvați un document ca pagină Web și apoi uitați-vă la el într-un editor de texte. Totul este o etichetă `<p>` urmată de o `class`. Clasele sunt utilizate apoi pentru a permite unei foi de stil imense să formateze codul. Funcționează, dar se ajunge la o sursă urâtă și dificil de citit.

Sugestie de la LeeAnne.com Words to weave by

Pe de altă parte, crearea de pagini Web generate de Word 2000 și apoi utilizarea XSL pentru a le transforma în XML standard sau chiar în XHTML, incluzând o foaie de stil, poate fi o metodă bună pentru a captura date moștenite pentru o aplicație Web.

Oricum, XML vă permite să etichetați acele secțiuni cu nume sugestive și vă permite încă să formatați acele secțiuni cu CSS. Sau cu orice altceva. Ultima parte este importantă. Pentru mine ca autor, acest lucru înseamnă că trebuie doar să pun o etichetă care să îi spună editorului meu ce înțeleg printr-o anumită linie de text. Pot lăsa un designer tipografic să decidă cum să o afișeze în modul cel mai bun și cel mai atractiv. Deci eu nu trebuie să știu nimic despre designul de carte, ci numai să scriu textul pe care îl va conține. Editorul meu nu trebuie să știe nimic despre XML, ci numai să interpreteze în context tipografic codurile pe care le-am scris.

După cum ați văzut în acest capitol, suportul existent pentru CSS și XSL este problematic. Generațiile următoare de browsere de la Netscape, Opera Software sau de la alți furnizori promit să îmbunătățească în mare măsură această situație. Dar problema suportului pentru browserele moștenite va mai exista pentru o bună bucată de vreme.

Depanarea CSS

Comenzile de stil nu funcționează așa cum ar trebui

Am ajustat o anumită proprietate text, iar acesta nu funcționează sau nu funcționează așa cum ar trebui. Uneori browserul meu se blochează atunci când utilizez o anumită comandă.

Bine ați venit în lumea captivantă a foilor de stil. În primul rând, încercați să validați foaia de stil utilizând instrumentul de validare de la W3C (`http://jigsaw.w3.org/css-validator/`) sau un altul. Validați apoi browserul dumneavoastră utilizând setul de teste W3C de la `http://www.w3.org/Style/css/Test/`. Poate veți cere un sfat de la cineva dacă ați obosit să vă dați cu capul de perete. Încercați CSS Pointers Group, la `http://css.nu/`.

V-ați săturat de foile de stil

De ce nu vor să funcționeze foile mele de stil atunci când le testez în alte browsere?

Dacă nu vă deranjează că documentul dumneavoastră seamănă remarcabil cu mii de alte situri de pe Web, primul lucru pe care trebuie să îl încercați este să utilizați foaia de stil a altcuiva. W3C păstrează cu bunăvoință, la îndemână, pe situl său Web o selecție de foi de stil testate. Ele testează browserul pe care îl utilizați și furnizează versiunea corectă, fără a fi nevoie de nici un efort din partea dumneavoastră. Uitați-vă la `http://www.w3.org/StyleSheets/Core/`. Dar dacă doriți ca oamenii să poată citi fără efort paginile dumneavoastră, atunci fiți atent să nu alegeți una dintre multele foi de stil cu fundal de culoare închisă. Este evident că specialiștilor Web de la Verso (`http://www.verso.com`) care au proiectat aceste stiluri le place acest efect, deoarece apare și pe pagina lor de bază. Însă acesta face paginile greu de citit dacă nu se mărește dimensiunea fontului.

O a doua alternativă este utilizarea unui limbaj de scriptare pentru a alege dintre câteva variante ale foii dumneavoastră de stil care funcționează pentru browsere diferite. În ciuda faptului că este mult de lucru pentru a realiza trei sau patru foi de stil pentru fiecare pagină, măcar este o operație realizabilă.

▶ Pentru informații despre validare și alte instrumente, **vezi** Anexa B, „Instrumente pentru editarea și transformarea XML/XHTML", la **pagina 678.**

În cele din urmă, XSL în sine va fi capabil să servească foi de stil diferite bazându-se pe negocierea conținutului cu browserul dumneavoastră cu suport XML. Dar această abilitate va trebui să aștepte până la următoarea versiune de XSL.

Depanarea foilor de stil XSL

Foile de stil nu fac ceea ce ar trebui

Am creat o foaie de stil grozavă care nu funcționează. Am încercat să o ajustez, dar tot degeaba.

În stadiul actual al lucrurilor, nu trebuie să concluzionați că dumneavoastră ați greșit. Presupunând că ați validat foaia dumneavoastră de stil împotriva erorilor simple, cu un instrument cum ar fi Stylus[tm], este posibil ca browserul dumneavoastră XML să nu suporte o anumită funcție pe care o utilizați. Cel mai bun lucru ar fi să întrebați pe cineva. Există un număr de liste poștale ai căror membri se ajută între ei cu sfaturi și chiar cu exemple scurte de cod. Pentru XSL, puteți încerca lista XSL găzduită de Mulberry Technologies. Vizitați situl lor Web pentru informații referitoare la abonare: `http://www.mulberrytech.com/`.

Selecțiile nu funcționează

Vreau să selectez un anumit nod din arborele documentului, dar nu reușesc.

Modelul document obiect (DOM) poate fi dificil de utilizat și de înțeles fără un studiu atent. Ați putea utiliza un instrument XML grafic care v-ar ajuta să aveți o vizualizare a structurii arborescente. Amintiți-vă, de asemenea, că dacă două șabloane XSL adresează același nod, atunci va fi utilizată potrivirea cea mai exactă.

Foi de stil în cascadă și XML/XHTML

CAPITOLUL 13

În acest capitol

- Utilizarea foilor de stil în cascadă
- Funcționarea foilor de stil
- Lupta cu implementările CSS din browsere
- Utilizarea elementelor XML ca selectori
- Selectori CSS
- Setarea perechilor CSS proprietate/valoare
- Depanare

Utilizarea foilor de stil în cascadă

În acest capitol presupuneți că foile de stil funcționează efectiv așa cum ar trebui, deoarece alternativa este de a concluziona că ele sunt atât de slab suportate încât ar fi efectiv inutile pe Web. Deși poate părea bizar, totuși are sens. Dacă sunteți atent, puteți crea foi de stil care nu se blochează. Iar în cel mai rău caz puteți utiliza script-uri pentru a alege foi de stil bazate pe probleme cunoscute.

Foile de stil în cascadă (CSS) sunt prima încercare a W3C de a separa considerentele de conținut de cele de afișare. Acest lucru permite autorului să aleagă ce fel de „decorație" să atârne de fiecare ramură a arborelui documentului. Ca și în cazul împodobirii bradului, cantitatea și tipul decorațiilor depind de dumneavoastră. Permițând ca deciziile de prezentare să fie luate separat de conținutul materialului, proiectantul este liber să creeze un aspect uniform pentru un sit, eliberându-i pe autori de grija aspectului informației după ce ajunge pe pagină. Până acum există două niveluri CSS, nivelul 3 fiind și el în curs de apariție. CSS1 nu se recomandă pe sine explicit pentru a fi utilizat la aplicarea unui stil în XML, dar CSS2 o face, deci cred că putem presupune că și motoarele CSS1 se vor putea referi la etichete XML, de dragul uniformității și compatibilității.

Veți utiliza CSS la un anumit nivel în viitorul apropiat, deoarece, după cum veți descoperi pe parcursul acestui capitol, XSL este mult mai complex și mult mai probabil să fie instrumentul specialiștilor. XSL și XSLT, pe de altă parte, au o putere extraordinară și pot fi utilizate pentru a efectua transformări ale afișării și redări complexe, dincolo de ceea ce poate face singur CSS.

Multe persoane care sunt obișnuite cu Web-ul și chiar cu proiectarea de situri Web complete cunosc prea puțin despre CSS, deoarece acesta nu prea funcționează cu combinația de browsere utilizate acum pe Web. Multe situri au luat decizia inteligentă de a nu uti-

liza deloc CSS, pentru a evita complexitatea și munca suplimentară în încercarea de a lua în considerare numeroasele probleme implicate de compatibilitate. Ca urmare, aici CSS este studiat mai detaliat decât ar fi necesar, acoperind atât ceva din fundamente, cât și punctele delicate.

De asemenea, foile de stil fac posibil ca un proiectant să țină cont de persoanele cu nevoi speciale. Accesibilitatea nu este limitată la persoane cu probleme de vedere sau alte handicapuri. Aceleași tehnici care sunt utilizate pentru utilizatori cu handicapuri fizice sau mentale sunt utile, în anumite circumstanțe, pentru întreaga populație. Deci șoferul unui automobil aflat în trafic are nevoie de un browser auditiv la fel de mult ca și o persoană cu un handicap de vedere, iar un mecanic auto care lucrează sub o mașină are nevoie de navigare hands-free (fără mâini) la fel de mult ca o persoană cu un handicap motor.

Funcționarea foilor de stil

Foile de stil funcționează prin identificarea unei etichete, a unei clase (numai în HTML), a unui atribut, ID sau a altei caracteristici structurate, identificabile, din arborele documentului, numită *selector* în CSS, și asociind selectorul cu o *declarație* conținând una sau mai multe perechi *proprietate/valoare*.

Foile de stil XML și XHTML ar trebui stocate întotdeauna extern, deoarece CSS utilizează o sintaxă de limbaj diferită atât de XML, cât și de XHTML, și ar fi interpretate greșit dacă nu ar fi împachetate într-o secțiune `CDATA` sau într-o instrucțiune de prelucrare. Mecanismul adecvat este utilizarea unei instrucțiuni de prelucrare pentru un document XML arbitrar. Pentru documentele XHTML există șase opțiuni non-exclusive:

- Utilizarea unui element `link`
- Utilizarea unei instrucțiuni de prelucrare `xml-stylesheet`
- Utilizarea unui element `style` XHTML cu o legătură externă
- Utilizarea unui element `style` cu un cod in-line transformat în comentariu
- Utilizarea de atribute `style` pentru elemente individuale
- Utilizarea de perechi in-line selector/declarație `style`, împachetate într-un element `CDATA`.

Totuși, transferarea de comenzi pentru foi de stil unui agent de afișare utilizând o comandă de prelucrare va bloca mulți agenți utilizator HTML existenți. Unii vor afișa incorect conținutul comenzii, dar aproape toți ceilalți o vor ignora. În același mod, agenții utilizator HTML existenți vor ignora conținutul `CDATA`, deci nici acest lucru nu ne este de ajutor. Cele două alternative externe, legături externe de la un element `link` sau `style`, nu sunt standard pentru XML, dar *ar trebui* să funcționeze și în XHTML. Totuși, foile de stil externe nu sunt suportate de multe browsere mai vechi. HTML 4.0 a definit conținutul `style` ca `CDATA`, deci întregul conținut era disponibil pentru utilizare chiar dacă era încadrat în comentarii. XHTML 1.0 definește de asemenea conținutul `style` ca `CDATA`, deci *ar trebui* să se comporte în același mod, singura excepție fiind aceea că analizoarele XML nu sunt obligate să transfere comentarii agentului utilizator, deci comentariile conținând informații despre stil pot fi extrase din ieșire, deci sunt invizibile pentru browser.

De fapt, în afara cazului în care creatorii de analizoare ignoră opțiunea de a elimina comentariile și permit browserului să vadă conținutul de stil comentat, XHTML nu este

teribil de util pe Web până când toți vechii agenți utilizator din lume vor dispărea în mod magic.

Deci, strategia pragmatică pare să fie încadrarea conținutului în comentarii pentru a acoperi și agenții moșteniți, utilizarea unei legături externe pentru a acoperi agenții actuali și viitori și speranța că va funcționa. Aceasta suportă cel mai larg număr de utilizatori actuali, fără a bloca browserele cu suport XML; totuși, nu se supune strict standardului XML.

Sugestie de la LeeAnne.com Words to weave by

Consider că renunțarea la mecanismul de comentare de dragul unui câștig imaginar în comportamentul *logic* a fost o greșeală imensă. În primul rând, Web-ul nu este deloc logic, deoarece abundă în incompatibilități și comportament discordant, tratamentul XML al comentariilor fiind doar unul dintre acestea. În al doilea rând, comportamentul de tip *eliminarea comentariilor* este opțional. De ce să vă încurcați cu blocarea potențială a aproape fiecărei pagini HTML existente care utilizează foi de stil, dacă nu aveți curaj suficient să vă urmați convingerile până la capăt?

Lupta cu implementările CSS din browsere

Ca să fim indulgenți, implementările CSS ale tuturor creatorilor importanți de browsere au fost neregulate. Deci, autorii vor economisi foarte mult timp necesar depanării dacă deciziile privind foile de stil vor fi lăsate pe seama unui specialist. Singurele strategii de stil posibile la momentul de față sunt fie furnizarea dinamică a unei foi de stil diferite pentru fiecare browser, fie simplificarea foii de stil până la punctul în care fiecare browser o afișează mai mult sau mai puțin corect. Există doar o mână de proprietăți care sunt sigure pentru a fi utilizate în această manieră.

Sugestie de la LeeAnne.com Words to weave by

Majoritatea autorilor ar trebui să se țină de instrumentele reale ale domeniului lor: titluri, paragrafe și diferite tipuri de liste. Crearea de stiluri este o operație specializată; este greu de învățat și ar trebui repartizată, dacă este posibil, unui expert în ciudățeniile implementării CSS pe diferite tipuri de browsere.

Generația următoare de browsere este posibil să îndulcească această problemă în unele privințe, dar rata de actualizare a browserelor este destul de scăzută, deci problema nu va dispărea curând. Deciziile regretabile și gafele flagrante făcute de implementatorii CSS de la producătorii importanți de browsere îi vor chinui mulți ani pe proiectanții de machete și formate de pagină.

Browserele Mozilla și Opera promit să conțină curând un suport aproape complet pentru CSS Level 2, deși istoria compatibilității CSS în Netscape este foarte săracă, iar Opera este până acum satisfăcător spre bun (în ciuda faptului că Opera este evident cel mai bun dintre browserele importante). Totuși, natura open-source a codului Mozilla (Netscape) înseamnă cel puțin că erorile și bug-urile inevitabile pot fi găsite și corectate, iar Opera a fost întotdeauna strâns legat de conformitatea cu standardele. Problema pare să fie că CSS este extrem de dificil de făcut corect.

Considerând că un browser Web este de fapt un motor de formatare pe loc, cum sunt cele din editoarele grafice (WYSIWYG), acest lucru nu ar trebui să fie o surpriză. Se pare că toată lumea are probleme cu efectuarea corectă a formatărilor grafice. Chiar și edi-

toarele cu mulți ani în spate au câteodată momente în care afișajul se întrerupe în moduri ciudate.

CSS poate modifica proprietățile documentului într-o ordine neliniară. Deci, tentativa dumneavoastră de machetare a paginii trebuie să ia în considerare, în mod dinamic, modificările din pseudo-elemente cum ar fi aliniat nou, alinierea față de ambele margini (poate) sau calcularea caracterelor care vor apărea efectiv pe prima linie, date fiind caracteristicile fizice ale dispozitivului dumneavoastră de afișare și dimensiunea ferestrei. Apoi, este posibil ca motorul de redare să fie nevoit să recalculeze totul atunci când treceți de la foaia de stil a documentului la foaia de stil a unui utilizator, modificând peste tot dimensiunea fontului. Iterațiile necesare pentru ca totul să fie corect și dificultatea operației înseamnă că nici unul din browserele importante nu are implementate caracteristici CSS *hard,* cum ar fi alinierea completă a textului la ambele capete.

Ultima implementare CSS a Microsoft, deși conține un suport limitat pentru CSS2 și XML, nu prea a îmbunătățit suportul CSS1. În ultima vreme, Microsoft a acordat mai multă atenție standardelor – cel puțin în publicitatea făcută – dar baza instalată de implementări Microsoft care nu sunt standard și conțin erori, printre care și CSS, este o responsabilitate imensă în încercarea de a evolua. Dacă includerea „compatibilității cu bug-urile anterioare" este o problemă la care trebuie să se gândească toți, în special Microsoft este foarte vulnerabil, deoarece majoritatea browserelor utilizate azi sunt de la Microsoft. Multe dintre ele sunt versiuni mai vechi, care probabil că nu vor fi actualizate niciodată până când calculatorul gazdă va fi aruncat și înlocuit. Ei sunt într-o situație dificilă, condamnați de unii utilizatori dacă o fac și condamnații de alții dacă nu o fac.

Utilizarea elementelor XML ca selectori

Elementele și atributele pot fi utilizate, teoretic, astfel încât să formeze *selectori* în CSS1 și CSS2, la fel ca în HTML. De fapt, CSS2 adresează direct această problemă păstrând numai notația *class* cu punct ca pe un caz HTML special. Selecția elementelor în CSS2 este asemănătoare cu cea din HTML, XHTML și din documentele XML generale. Deci dacă evitați notația cu punct, aceleași foi de stil pot fi folosite atât pentru documentele XML cât și pentru cele HTML, schimbând pur și simplu numai numele elementelor.

Totuși, multe browsere au încercat să își păstreze cunoștințele speciale HTML pentru a putea face presupuneri rezonabile despre modul în care să corecteze pe loc erorile din pagină. Aceasta înseamnă că browserele nu înțeleg deloc sensul CSS, dar *simulează* compatibilitatea cu CSS printr-o serie de rutine cu scop special. În timp ce așa-numitele lor motoare CSS sunt slabe pentru HTML, pentru documentele XML generale sunt aproape inutile, deoarece nu există nici o metodă de a crea cunoștințe speciale pentru nume de elemente arbitrare. În ciuda faptului că HTML avea definit un DTD aproape din primele sale zile, implementările actuale îl consideră oarecum consultativ, dacă nu cumva îl ignoră, soluțiile rudimentare și improvizațiile încercând să compenseze renumita stare proastă a designului Web și mizeria HTML larg răspândită, dar mascată în pagini utilizabile. Dar, evident, după ce permiteți explicit o sintaxă diformă și greșeli comune, practicile de codare defectuoasă proliferează fără limite.

Selectori CSS

Numeroasele proprietăți și valori CSS nu sunt chiar așa de greu de înțeles. Pentru cei mai mulți, ele sunt destul de logice, cel puțin pentru vorbitorii de limba engleză, în ciuda faptului că învățarea pe dinafară a cuvintelor exacte vă poate speria. O prezentare completă a XHTML apare în Anexa A, „Referințe XML/XHTML", iar a proprietăților CSS1 și CSS2 în Anexa D, „Referințe CSS1, CSS2", deci nu vă voi plictisi cu ele aici. Se presupune că aveți deja un minimum de cunoștințe despre foile de stil în HTML.

În esență, *elementele* se referă la etichete din documentul dumneavoastră, iar *pseudo-elementele* se referă la etichetele virtuale, care nu există *efectiv* în document. Browserul utilizează aceste etichete reale și etichete virtuale pentru a decide când și unde să aplice proprietățile și valorile trecute ca argumente în element.

Alcătuirea foilor de stil este guvernată de reguli. Tabelul 13.1 rezumă modul în care sunt asamblate etichetele, atributele și valorile pentru a forma un selector complet:

Tabelul 13.1 Selectori CSS

Model	Semnificație
`*`	Se referă la orice element
`E`	Se referă la orice element `E`
`E F`	Se referă la orice element `F`, descendent al unui element `E`
`E > F`	Se referă la orice element `F`, copil al unui element `E`
`E:first-child`	Se referă la orice element `E`, atunci când acesta este un prim copil
`E:link`	Se referă la orice element `E`, dacă `E` este ancora sursă a unei hiperlegături atunci când ținta nu a fost încă vizitată
`E:visited`	Se referă la orice element `E`, dacă `E` este ancora sursă a unei hiperlegături atunci când ținta a fost deja vizitată
`E:active`	Se referă la orice `E`, pe durata anumitor acțiuni ale utilizatorului
`E:hover`	Se referă la orice `E`, pe durata anumitor acțiuni ale utilizatorului
`E:focus`	Se referă la orice `E`, pe durata anumitor acțiuni ale utilizatorului
`E:lang(c)`	Se referă la orice element de tipul `E`, dacă acesta este în limba (umană) `c`
`E + F`	Se referă la orice element `F` având ca frate imediat precedent un element `E`
`E[foo]`	Se referă la orice element `E` care are setat atributul `foo` (indiferent de valoare)
`E[foo="warning"]`	Se referă la orice element `E` al cărui atribut `foo` are chiar valoarea `"warning"`
`E[foo-="warning"]`	Se referă la orice element `E` al cărui atribut `foo` are ca valoare o listă de valori separate prin spații, dintre care una este exact `"warning"`
`E[lang|="en"]`	Se referă la orice element `E` al cărui atribut `lang` are o listă de valori separate prin cratime, care începe (de la stânga) cu `"en"`
`DIV.warning`	*Numai pentru HTML* – identic cu `DIV[class-="warning"]`
`E#myid`	Se referă la orice element `E` având ID egal cu `myid`

Observați că sintaxa class cu punct este singurul model favorizat de HTML și nu poate fi utilizată cu siguranță în XML,în ciuda părerii contrare a unora. Aceasta deoarece, pentru XML, class este doar un alt atribut. Deoarece puteți utiliza sintaxa explicită valoare atribut pentru a realiza același lucru, nu se pierde decât comoditatea unui format prescurtat.

Pe de altă parte, un ID XML are o poziție privilegiată, deși este posibil să nu aibă acest nume. Un ID XML trebuie să aibă o valoare de „ID" simbolică, dar poate avea orice nume, deși într-un element poate exista un singur „ID".

În utilizare, se pot folosi selectori pentru a construi comenzi pentru foi de stil, așa cum se arată și în secțiunea următoare.

Utilizarea selectorilor pentru a construi comenzi pentru foi de stil

Selectorul următor selectează elementele linesegment dintr-un document XML, permițându-vă să definiți una sau mai multe perechi proprietate/valoare pentru toate instanțele acelui element:

```
linesegment { property: value; property: value; … }
```

Selectorul următor selectează elementele h1, h2 și h3 ale unui document XML sau HTML, permițându-vă să definiți una sau mai multe perechi proprietate/valoare pentru toate instanțele celor trei elemente:

```
h1, h2, h3 { property: value; property: value; … }
```

Selectorul următor selectează orice element paragraph, descendent al unui element chapter dintr-un document XML, permițându-vă să definiți una sau mai multe perechi proprietate/valoare pentru acele instanțe ale acestui element:

```
chapter paragraph { property: value; property: value; … }
```

Selectorul următor selectează orice element paragraph având ca părinte un element chapter dintr-un document XML, permițându-vă să definiți una sau mai multe perechi proprietate/valoare pentru acele instanțe ale acestui element:

```
chapter > paragraph { property: value; property: value; … }
```

Selectorul următor selectează primul element paragraph având ca părinte un element chapter dintr-un document XML, permițându-vă să definiți una sau mai multe perechi proprietate/valoare pentru acele instanțe ale acestui element. Acesta poate fi utilizat, de exemplu, pentru a defini un tratament special al primului paragraf dintr-un capitol:

```
chapter:first-child paragraph { property: value; property: value; … }
```

Selectorul următor selectează prima literă din primul element paragraph având ca părinte un element chapter dintr-un document XML, permițându-vă să definiți una sau mai multe perechi proprietate/valoare pentru acele instanțe ale acestui element. Acesta poate fi utilizat, de exemplu, pentru a defini o letrină inițială, cu rol decorativ, pentru primul paragraf dintr-un capitol:

```
chapter:first-child paragraph:first-letter { property: value; property: value; … }
```

Intervalul de selectori posibili este aproape infinit, deoarece ei pot fi combinați în aproape orice mod. Pentru mai multe referințe, vedeți Recomandările W3C sau un text specializat în foile de stil în cascadă.

Utilizarea pseudo-elementelor și a pseudo-claselor

Un număr de pseudo-elemente și pseudo-clase nu au o realitate obiectivă în codul sursă al paginii. Ele există pentru a face posibile, fără utilizarea unor etichete speciale, tratamente tipografice speciale asupra textului. Multe dintre aceste caracteristici necesită informații de la browser și nu de la autor, deoarece deciziile finale pot fi luate doar în momentul în care pagina este redată.

De exemplu, pseudo-elementul first-line (prima linie) permite autorului să trateze prima linie de text dintr-un paragraf (sau alt element bloc) în mod diferit față de restul corpului de text. Exemple obișnuite ar fi: aliniat nou; afișarea primei linii cu aldine, cu majuscule reduse, sau alt tratament special pentru text; sau chiar o ușoară modificare a dimensiunii fontului. Alcătuirea exactă a primei linii diferă în funcție de font, dimensionea fontului, lungimea liniei și alte variabile. Proiectantul Web are, în cel mai bun caz, doar un control minimal asupra acestor lucruri. Și așa și trebuie. Majoritatea acestor decizii ar trebui lăsate în seama nevoilor sau dorințelor utilizatorului.

Ar trebui reținut faptul că în Tabelul 13.2 și în Tabelul 13.3, prima linie (first-line) și prima literă (first-letter) pot fi atașate doar elementelor la nivel de bloc și pot afecta numai anumite proprietăți. În plus, pseudo-clasele link, active și visited pot fi atașate doar elementului A în CSS1, deși extinderea acestei funcționalități (și a altora) la elementele XML este obișnuită, dar nu este standard. Tabelele următoare prezintă exemple generale de selectori, unde X reprezintă un element arbitrar.

Tabelul 13.2 Pseudo-elemente CSS1

Pseudo-element	Descrie
X:link	Legătură nevizitată
X:visited	Legături vizitate
X:active	Legături active
X:first-letter	Prima literă a unui element bloc
X:first-line	Prima linie a unui element bloc

Tabelul 13.3 Pseudo-clase CSS2

Pseudo-clasă	Descrie
X:first-child	Primul element copil al unui element X
X:link	Legătură nevizitată a unui element link X
X:visited	Legătură vizitată a unui element link X
X:hover	Acțiune realizată de utilizator
X:active	Legătură activă a unui element link X
X:focus	Acțiune realizată de utilizator
X:lang(lang-code)	Corespunde unui atribut de limbă
X:first-letter	Prima literă a unui element bloc
X:first-line	Prima linie a unui element bloc
X:before	Accesează locația imediat anterioară unui element, deci poate fi inserat conținut în acel punct
X:after	Accesează locația imediat următoare unui element, deci poate fi inserat conținut în acel punct

Am văzut deja mai multe exemple de utilizare a pseudo-elementelor și pseudo-claselor în secțiunea anterioară „Selectori CSS", deci nu le vom mai repeta aici.

Setarea perechilor CSS proprietate/valoare

După ce un element a fost selectat îi puteți aplica o pereche proprietate/valoare. Lista proprietăților este foarte mare, fapt confirmat și de o privire fugară asupra Anexei D. Multe din efectele interesante sunt disponibile numai în CSS2, iar extensiile planificate pentru CSS3 vă vor permite un control chiar mai mare asupra aspectului și machetei documentului. Atunci când aceste niveluri vor fi implementate în fiecare browser XML, viața ar trebui să devină interesantă.

▶Pentru mai multe informații despre foile de stil în cascadă, **vezi** „Referință rapidă CSS" la **pagina 710.**

Exemplele următoare descriu proprietăți de redare simple care pot fi asociate cu elementele XML utilizând perechi selector XML, declarație, cu un text descriptiv:

```
paragraph { font-size: 14pt; }
```

Setează dimensiunea fontului unui element `paragraph` la 14 puncte.

```
paragraph + sub-paragraph { text-indent: 0; }
```

Dacă un element `subparagraph` este imediat precedat de fratele său `paragraph`, atunci nu îl indentează.

```
* [status="deleted"] { text-decoration: line-through; }
```

Dacă orice element are un atribut `status` cu valoarea egală cu `"deleted"`, atunci taie cu o linie orice text conținut.

```
division > paragraph:first-child { text-indent: 5pt; }
```

Dacă un element `paragraph` este primul copil al unui element `division`, atunci setează indentul la 5 puncte.

Cu același selector pot fi setate mai multe perechi proprietate/valoare prin simpla lor trecere între acolade. Ultimul caracter punct și virgulă are rol de separator și ar trebui inclus întotdeauna.

Clase CSS

Deși selectorul de clasă nu este utilizat în XML, ar trebui să fiți capabil să îl recunoașteți pentru a putea converti ușor notația la o alternativă mai sigură:

```
div.warning { color: red; }
```

Într-un document HTML, dacă un element `div` are un atribut `class` cu o valoare egală cu `"warning"`, atunci culoarea textului este setată la `"red"` (roșu).

Acest lucru este interzis în XML, deoarece un atribut `class` nu are nici o semnificație specială. Dar același efect îl puteți obține astfel:

```
div[class="warning"] { color: red; }
```

În orice document HTML sau XML, dacă un element `div` are un atribut `class` cu o valoare egală cu `"warning"`, atunci culoarea textului este setată la `"red"`.

Din toate conversiile existente în lume, aceasta este printre cele mai simple. Puteți scrie cu ușurință un script Perl care să facă acest lucru pentru dumneavoastră dacă aveți de efectuat multe conversii, de exemplu atunci când transformați un sit întreg pentru a fi compatibil XML și nu știți cu exactitate unde se află astfel de construcții.

Utilizarea ID-urilor XML și HTML

ID-urile au o poziție specială atât în XML cât și în HTML, deși există o mică incompatibilitate în modul în care sunt speciale. În HTML și XML, sintaxa de stil este

```
x#value { property: value; }
```

unde `x` poate fi orice element, iar `value` este valoarea unui atribut numit `id` și setează o proprietate oarecare la o valoare oarecare.

Din păcate, această sintaxă comodă este utilizată și într-un URI pentru a identifica o poziție într-un document, numită uneori și *fragment*. Totuși, ținta descrisă de această sintaxă în HTML este un element cu un atribut `name` setat la acea valoare. Pe de altă parte, în XML ținta este un element cu un atribut `id` având acea valoare. Deci, în XHTML, la fel ca în orice document XML, cea mai bună metodă este probabil utilizarea de fiecare dată a ambelor nume de atribute și setarea lor la aceeași valoare. Următorul exemplu este în XHTML:

```
<paragraph id="target1" name="target1">Informatii</paragraph>
```

În acest fel, ținta este găsită indiferent de convenția urmată de agentul utilizator. În XML, ID-ul trebuie să fie unic în document, în caz contrar documentul fiind incorect. În HTML ambele atribute, `name` și `id`, trebuie să fie unice, dar de obicei nu sunt verificate. În XML, ID-ul poate avea orice nume. Deci,

```
<paragraph identifier="target1" name="target1">Informatii</paragraph>
```

conține un „ID" dacă DTD-ul arată (parțial) astfel:

```
<!ELEMENT paragraph (#PCDATA)>
<!ATTLIST paragraph identifier ID       #REQUIRED
                    name       NMTOKEN  #REQUIRED >
```

Deci, în XML, dacă doriți ca un ID să aibă efectiv *numele* `"id"`, va trebui să îl declarați în DTD astfel:

```
<!ELEMENT paragraph (#PCDATA)>
<!ATTLIST paragraph id   ID       #REQUIRED
                    name NMTOKEN  #REQUIRED >
```

Tipuri de mediu

CSS2 a introdus conceptul de tipuri diferite de mediu, adică dispozitivele țintă care partajează anumite caracteristici identificabile. Deși au nume specifice, ar trebui reținut faptul că toate dispozitivele care partajează același set de caracteristici sunt de același tip de mediu. Pe măsură ce CSS evoluează și apar noi dispozitive, vor putea fi definite noi tipuri de mediu. Un tip de mediu CSS numește un set de proprietăți CSS. Un agent utilizator care pretinde că suportă după nume un tip de mediu trebuie să implementeze toate proprietățile care se aplică acelui tip de mediu.

Notă

Termenul *mediu*, atunci când este utilizat în CSS, se referă la clasa generală de dispozitive de ieşire, cum ar fi „handheld" care se referă la dispozitive de ieşire mici: telefoane celulare şi asistenţi digitali personali, precum şi calculatoarele hand-held specializate, utilizate deseori pentru inventarieri şi aplicaţii pentru colectare de date. Analog, „print" se referă în general la un mediu paginat, atât pentru ieşire pe hârtie, cât şi pentru dispozitive de afişare care emulează o bucată de hârtie virtuală prin constrângerea redării la o pagină fără defilare.

Numele alese pentru tipurile de mediu CSS reflectă dispozitivele ţintă pentru care au sens proprietăţile relevante. În lista de tipuri de mediu CSS din Tabelul 13.4, descrierile trecute între paranteze nu sunt normative. Ele specifică doar cărui dispozitiv îi este destinat tipul de mediu respectiv.

Tabelul 13.4 Tipuri de mediu CSS

Tip de mediu	Mediu ţintă
`all`	Potrivit pentru toate dispozitivele.
`aural`	Destinat pentru sintetizatoarele de voce. Pentru mai multe detalii, vezi secţiunea „Redarea auditivă".
`Braille`	Destinat pentru dispozitivele de afişare Braille.
`embossed`	Destinat pentru imprimantele Braille.
`handheld`	Destinat pentru dispozitivele hand-held (de obicei cu ecran mic, monocrom şi lăţime de bandă limitată).
`print`	Destinat pentru materiale paginate şi opace şi pentru documente vizualizate pe ecran în modul print preview (previzualizare înainte de tipărire). Pentru informaţii referitoare la problemele de formatare specifice unui mediu paginat, vă rog să consultaţi secţiunea „Suport şi emulare pentru copii tipărite CSS3".
`projection`	Destinat pentru prezentările proiectate, de exemplu cu retroproiectoare sau prin imprimare pe folii transparente. Pentru informaţii referitoare la problemele de formatare specifice unui mediu paginat, vă rog să consultaţi secţiunea „Suport şi emulare pentru copii tipărite CSS3".
`screen`	Destinat în principal pentru monitoarele color.
`tty`	Destinat pentru afişaje care folosesc caractere de lăţime fixă, cum sunt teleimprimatoarele, terminalele şi dispozitivele portabile cu capacităţi de afişare limitate. Autorii nu ar trebui să folosească unităţi în pixeli la tipul de mediu `tty`.
`tv`	Destinat pentru dispozitivele de tip televizor (ecrane cu rezoluţie mică, color, posibilitate de defilare limitată şi sunet).

Numele tipurilor de mediu nu sunt sensibile la tipul literelor.

Tipurile de mediu merg mână în mână cu regula `@media`. O regulă `@media` defineşte o secţiune a unei foi de stil, delimitată prin acolade, care se aplică unui anumit mediu.


```
@media print {
        BODY { font-size: 10pt }
        }
@media screen {
        BODY { font-size: 12pt }
        }
@media screen, print {
        BODY { line-height: 1.2 }
        }
```

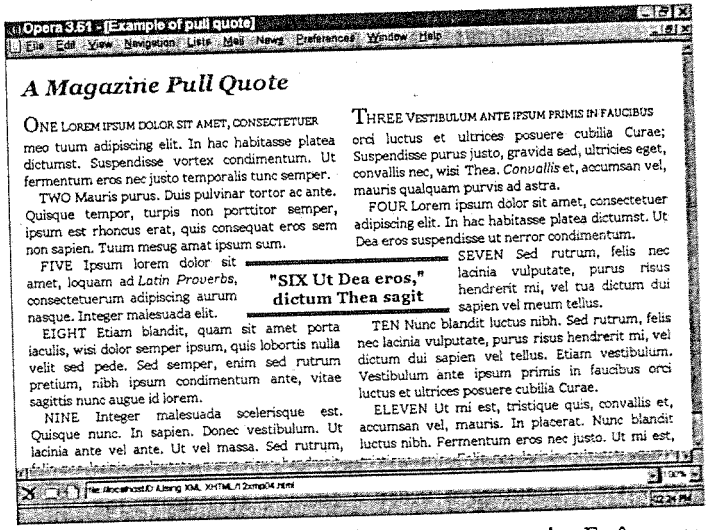
În fiecare din exemplele anterioare, în fiecare regulă de mediu puteau fi definiţi selectori multipli şi perechi proprietate/valoare.

Suport şi emulare pentru copii tipărite CSS3

Ieşirea copie tipărită şi ieşirea copie tipărită simulată pe ecran sunt invenţii noi în CSS3. Multe efecte de text importante au sens numai atunci când sunt limitate la o singură pagină de dimensiuni cunoscute. Utilizarea frecventă a textului pe două coloane, având o casetă centrală „citat evidenţiat", întâlnită des în reviste, este un astfel de exemplu. În ciuda faptului că cineva poate simula acest lucru prin multă muncă utilizând HTML/XHTML, efectul nu se păstrează prea bine după editări sau la redimensionarea dinamică a ferestrei. Figura 13.1 prezintă un exemplu de pagină cu un citat evidenţiat. Observaţi codul asociat care nu este liniar şi va necesita o remodelare atentă dacă veţi dori să adăugaţi cuvinte la o secţiune.

FIGURA 13.1

Observaţi că acesta este un tur de forţă în Opera – cu citatul din centru simulat manual, în loc să fie descris simplu. Ideea este că CSS3 suportă medii paginate şi va fi capabil să facă acest lucru în mod direct, utilizând obiecte mobile bazate pe pagini.



Listingul 13.1 prezintă foaia de stil care face să funcţioneze acest exemplu. Ea împarte pagina în secţiunile stânga şi dreapta, pe care le împarte apoi în secţiunile sus, mijloc şi jos, stânga şi dreapta, pentru a adăposti citatul evidenţiat care are propriii selectori de stil şi propriile proprietăţi. În plus, ea împarte antetul în secţiunile stânga şi dreapta, pentru ca totul să se potrivească.

Listingul 13.1 Foaie de stil care simulează o casetă de text mobilă

```
EM { font-style: italic; }
BODY { background: none #ffffff; color: #000000; font-size: 1em;
       font-family: Georgia,"Goudy Old Style","Bookman Old Style","Times New
➥Roman",serif;
       font-weight: normal; margin: 0; border: none; padding: 0;
       width: auto; }
DIV.lefthead { background: #ffffff url(../graph/pmarblet.jpg) top left
               repeat-x;  color: #000000; margin: 0 80px 0 0; border: none;
               padding: 0; height: 50px; width: auto; }
DIV.righthead { background: #ffffff url(../graph/JR-logo.jpg) top right
                no-repeat; color: #000000; float: right; margin: 0;
                border: none; padding: 0; height: 50px; width: 80px; }
DIV.left1 { margin: 0; border: none; padding: 0; float: left; width: 50%; }
DIV.right1 { margin: 0; border: none; padding: 0; float: right; width: 50%; }
DIV.left2 { margin: 0; border: none; padding: 0; float: left; width: 35%; }
DIV.right2 { margin: 0; border: none; padding: 0; float: right; width: 35%; }
DIV.generic { margin: 0; border: none; padding: 0; width: auto; }
DIV.pullquote { margin: 0.6em 0 0 0; border-top: double #000000 5px;
                border-right: none; border-bottom: double #000000 5px;
                border-left: none; padding: 0; float: left; width: 30%; }
H1 { background: transparent; color: #000000; font-size: 1.6em;
             font-family: Georgia,"Goudy Old Style","Bookman Old Style","Times
➥New Roman",serif;
             font-weight: bold; font-style: italic; margin: 0; border: none;
             padding: 20px 0 0 20px; text-align: left; width: auto; }
H2 { background: #ffffff; color: #000000; font-size: 1.4em;
               font-family: Georgia,"Goudy Old Style","Bookman Old Style","Times
➥New Roman",serif;
               font-weight: bold; margin: 0; border: none;
               padding: 0.3em; text-align: center; width: auto; }
IMG { float: right; margin: 0.4em 0 0 0; border: none; padding: 0; }
P { margin: 0; border: none; background: #ffffff; color: #000000;
    font-family: Georgia,"Goudy Old Style","Bookman Old Style","Times New
➥Roman",serif;
    font-size: 1em; font-weight: normal; text-align: justify;
    line-height: 1.3; }
P.firstleft:first-letter { font-size: 1.6em; }
P.firstleft:first-line { font-variant: small-caps; }
P.firstleft { padding: 0.7em 0.6em 0 20px; text-indent: 0; }
P.firstright:first-letter { font-size: 1.6em; }
P.firstright:first-line { font-variant: small-caps; }
P.firstright { padding: 0.7em 20px 0 0.6em; text-indent: 0; }
P.bodyleft { padding: 0 0.6em 0 20px; text-indent: 1em; }
P.bodyright { padding: 0 20px 0 0.6em; text-indent: 1em; }
P.spacer { padding: 0; clear: both; font-size: 0em; width: auto; }
```

Listingul 13.2 prezintă codul XHTML. Observați că următorul cod separă textul care trebuie plasat în coloanele virtuale în blocuri disjuncte și nesecvențiale. Aceasta pentru ca totul să *apară* într-o dispunere naturală, asemenea unei coloane de revistă sau ziar, în ciuda faptului că browserele actuale nu suportă acest lucru. Ar trebui menționat faptul

că versiunea beta a browserului Opera suportă medii paginate, deci posibilitatea de a defini în mod real acest lucru nu este chiar atât de îndepărtată.

Listingul 13.2 Cod sursă XHTML care simulează o casetă de text mobilă

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/strict.dtd">
<html
 lang="en-US"
 xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      Example of pull quote
    </title>
    <link
     rel="StyleSheet"
     href="12xmp04.css"
     type="text/css"
     media="screen" />
  </head>
  <body>
    <div class="righthead">
      <p class="spacer">
      </p>
    </div>
    <div class="lefthead">
      <h1>
         A Magazine Pull Quote
      </h1>
    </div>
    <div class="left1">
      <p class="firstleft">
         ONE Lorem ipsum dolor sit amet, consectetuer meo tuum
         adipiscing elit. In hac habitasse platea dictumst.
         Suspendisse vortex condimentum. Ut fermentum eros nec justo
         temporalis tunc semper.
      </p>
      <p class="bodyleft">
         TWO Mauris purus. Duis pulvinar tortor ac ante. Quisque
         tempor, turpis non porttitor semper, ipsum est rhoncus
         erat, quis consequat eros sem non sapien. Tuum mesug amat
         ipsum sum.
      </p>
    </div>
    <div class="right1">
      <p class="firstright">
         THREE Vestibulum ante ipsum primis in faucibus orci luctus
         et ultrices posuere cubilia Curae; Suspendisse purus justo,
         gravida sed, ultricies eget, convallis nec, wisi Thea. <em>
         Convallis</em> et, accumsan vel, mauris qualquam purvis ad
         astra.
      </p>
```

Listingul 13.2 (continuare)

```
    <p class="bodyright">
       FOUR Lorem ipsum dolor sit amet, consectetuer adipiscing
       elit. In hac habitasse platea dictumst. Ut Dea eros
       suspendisse ut nerror condimentum.
    </p>
  </div>
  <p class="spacer">
  </p>
  <div class="left2">
    <p class="bodyleft">
       FIVE Ipsum lorem dolor sit amet, loquam ad <cite>Latin
       Proverbs</cite>, consectetuerum adipiscing aurum nasque.
       Integer malesuada elit lauditum erat meum sit.
    </p>
  </div>
  <div class="pullquote">
    <h2>
       "SIX Ut Dea eros," dictum Thea sagit
    </h2>
  </div>
  <div class="right2">
    <p class="bodyright">
       SEVEN Sed rutrum, felis nec lacinia vulputate, purus risus
       hendrerit mi, vel tua dictum dui sapien vel meum noventum
       eram nic qua perditat tellus.
    </p>
  </div>
  <p class="spacer">
  </p>
  <div class="left1">
    <p class="bodyleft">
       EIGHT Etiam blandit, quam sit amet porta iaculis, wisi
       dolor semper ipsum, quis lobortis nulla velit sed pede. Sed
       semper, enim sed rutrum pretium, nibh ipsum condimentum
       ante, vitae sagittis nunc augue id lorem.
    </p>
    <p class="bodyleft">
       NINE Integer malesuada scelerisque est. Quisque nunc. In
       sapien. Donec vestibulum. Ut lacinia ante vel ante. Ut vel
       massa. Sed rutrum, felis nec lacinia vulputate, purus risus
       hendrerit. Integer malesuada scelerisque est.
    </p>
  </div>
  <div class="right1">
    <p class="bodyright">
       TEN Nunc blandit luctus nibh. Sed rutrum, felis nec
       lacinia vulputate, purus risus hendrerit mi, vel dictum dui
       sapien vel tellus. Etiam vestibulum. Vestibulum ante ipsum
       primis in faucibus orci luctus et ultrices posuere cubilia
       Curae.
    </p>
```



```
    <p class="bodyright">
       ELEVEN Ut mi est, tristique quis, convallis et, accumsan
       vel, mauris. In placerat. Nunc blandit luctus nibh.
       Fermentum eros nec justo. Ut mi est, tristique quis. Felis
       nec lacinia vulputate, purus risus hendrerit mi, vel dictum
       vel tellus.
    </p>
  </div>
 </body>
</html>
```

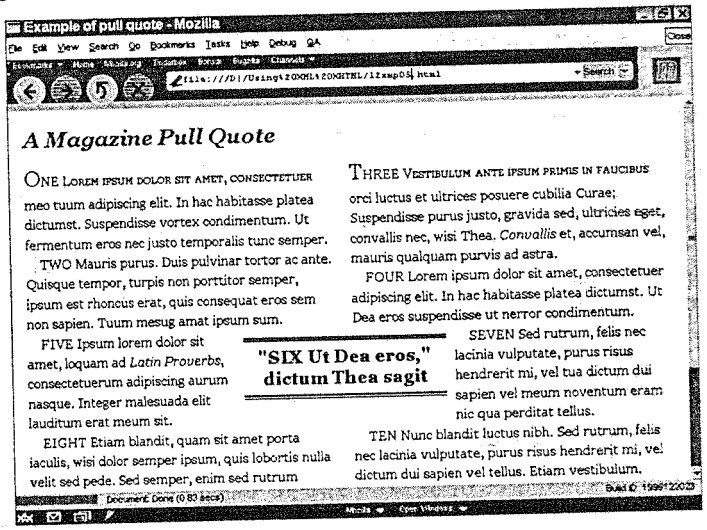
În locul unei instrucţiuni de prelucrare a fost utilizat un element `link`, deoarece această sintaxă este cel mai bine înţeleasă. În ciuda sfatului meu, pentru a economisi spaţiu foaia de stil nu a fost utilizată in-line.

De asemenea, a fost utilizată notaţia `class` cu punct deoarece Opera, singurul browser în afară de Mozilla care înţelege această foaie de stil, nu înţelege momentan notaţia cu paranteze.

Figura 13.2 prezintă pagina în browserul Mozilla.

FIGURA 13.2

Observaţi că marginea citatului evidenţiat în Mozilla nu este prea aspectuoasă. Ar fi putut să arate mai bine în Mozilla şi puţin mai urât în Opera, dar totul este un compromis.



Pagina nu este prezentată în Microsoft Internet Explorer sau Netscape Navigator 4.7, deoarece în acestea nu funcţionează deloc. Nu este o surpriză, deoarece în cazul în care foaia de stil trece prin validatorul de foi de stil TopStyle, acesta vă va avertiza despre acest lucru din cauza bug-urilor cunoscute din browserele importante.

Redarea auditivă

Unul dintre punctele forte ale lui CSS2 este aptitudinea sa de a defini foi de stil diferite pentru medii diferite. Deoarece dispozitivele de redare auditivă au caracteristici complet diferite faţă de cele vizuale, caracteristicile speciale ale browserelor auditive pot fi luate în considerare fără a afecta celelalte dispozitive. Pe situl Web Henter-Joyce, la

http://www.hj.com, sunt disponibile pentru descărcare fișiere multimedia care descriu cititorul de ecran JAWS, precum și o versiune demonstrativă a programului și o prezentare a modului în care JAWS poate interacționa cu aplicația lor de mărire a ecranului, MAGic, pentru a crea medii de lucru adaptabile pentru persoane cu vederea slabă sau cu alte nevoi speciale.

Pentru a face distincție între diferiți vorbitori, un browser auditiv îi poate plasa într-o poziție simulată spre stânga, respectiv spre dreapta. Accentuările pot fi realizate prin modificarea volumului, iar titlurile pot fi distinse după frecvență. Unele browsere, sau cel puțin hardware-ul lor fizic, au și capacitatea de a seta diferite voci și de a utiliza poziții pe verticală simulate, deasupra sau dedesubtul nivelului urechii utilizatorului. Deci, lista de proprietăți proiectată pentru aceste browsere poate părea ciudată la început.

Fragmentul următor de foaie de stil prezintă utilizarea proprietăților auditive pentru redarea unui document XML.

```
{ volume: <number> | <percentage> | silent |
          x-soft | soft | medium | loud |
          x-loud | inherit ; }
```

Setează volumul.

```
{ speak: normal | none | spell-out | inherit ; }
```

Setează stilul vorbirii.

```
{ pause-before: <time> | <percentage> | inherit ; }
```

Setează pauza dinaintea vorbirii. Aceasta oferă ascultătorului timp să se pregătească pentru semnalul auditiv sau să realizeze unele acțiuni necesare, cum ar fi așezarea mâinilor pe tastatură într-o poziție odihnitoare după acționarea unei taste de control.

```
{ pause-after: <time> | <percentage> | inherit ; }
```

Setează pauza de după vorbire. Aceasta ajută ascultătorul să facă distincție între idei separate, deci cineva poate să o utilizeze pentru o pauză după propoziții și o pauză puțin mai lungă după paragrafe.

```
{ pause: <time> | <percentage> | inherit ; }
```

Proprietate prescurtată pentru setarea pauzei dinainte și de după vorbire. Logic, poate fi utilizată pentru a pune în paranteză declarațiile, cu o pauză la fiecare capăt pentru a scoate în evidență, de exemplu, o frază între paranteze față de textul înconjurător.

```
{ cue-before: <url> | none | inherit ; }
```

Setează un semnal sonor de atenționare. Acesta poate fi utilizat pentru a emite un sunet înaintea unui răspuns sau a unui mesaj de avertizare.

```
{ cue-after: <url> | none | inherit ; }
```

Setează un semnal sonor de atenționare. Acesta poate fi utilizat, de exemplu, pentru a semnala utilizatorului să realizeze o anumită acțiune, cum ar fi oprirea unui alt dispozitiv de redare înainte de a continua.

```
{ cue: <url> | none | inherit ; }
```

Proprietate prescurtată pentru setarea semnalelor sonore de atenționare prezentate mai sus. Această proprietate poate fi utilizată, de exemplu, pentru a introduce și pentru a încheia în text comentarii aflate între paranteze.


```
{ play-during: <url> | mix? | repeat? | auto
               none | inherit ; }
```

Setează un sunet care să fie redat în timpul altui sunet, deci poate fi creată o prezentare auditivă neliniară fără a avea suport SMIL. Un exemplu des întâlnit pe Web este muzica de fundal a unor pagini.

```
{ azimuth: <angle> | left-side | far-left | left |
           center-left | center | center-right |
           right | far-right | right-side | behind |
           leftwards | rightwards | inherit ; }
```

Setează locația simulată a unei surse de sunet sau a unui vorbitor. Observați că obținerea întregului complet de efecte posibile necesită un fel de sistem audio surround, deși sistemul stereo este adecvat pentru o plasare simplă stânga-dreapta a sursei.

```
{ elevation: <angle> | below | level | above | higher |
             lower | inherit ; }
```

Setează poziția pe verticală simulată a unei surse de sunet sau a unui vorbitor. elevation este măsurată într-un sens pozitiv din față și de la nivelul ascultătorului, apoi deasupra, în spatele său, apoi dedesubt, și iar în față. În sens negativ, locația percepută se mută din față, dedesubt, în spate, deasupra, apoi iar în față. Rețineți că unghiurile de plasament vertical al sunetelor necesită mai multe canale și boxe decât pot fi asigurate de sistemele audio actuale. Pare probabil că sistemele pentru jocuri vor oferi această posibilitate înainte ca ea să devină larg disponibilă în alte locuri.

```
{ speech-rate: <number> | x-slow | slow | medium |
               fast | x-fast | faster | slower | inherit ; }
```

Setează ritmul vorbirii. number reprezintă cuvinte pe minut.

```
{ voice-family: <specific voice> | <generic voice> |
                inherit ; }
```

Setează o familie de voci. Vocile specifice depind de agentul utilizator.

```
{ pitch: <frequency> | x-low | low | medium | high |
         x-high | inherit ; }
```

Setează înălțimea medie a vorbirii (frecvența). Bărbat = 120 Hz, Femeie = 210 Hz, iar Copil = 300 Hz.

```
{ pitch-range: <number> | inherit ; }
```

Setează intervalul de înălțime, modulația medie pe care un vorbitor o utilizează într-o propoziție. Vocile feminine tind să utilizeze un interval de înălțime mai larg, iar vocile bărbătești tind spre un interval de înălțime mai îngust. Numărul este între 0-100. Media = 50.

```
{ stress: <number> | inherit ; }
```

Setează modelul de intonație. Numărul este între 0-100.

```
{ richness: <number> | inherit ; }
```

Setează modelul de nuanță al unei voci, de la monoton la foarte bogat. Numărul este între 0-100.

```
{ speak-punctuation: code | none | inherit ; }
```

Setează punctuația vorbită sau dedusă din pauze și ton.

BIBL. CENTRALĂ UNIV. „AL. I. CUZA" IAȘI SECȚIA DE INFORMARE

```
{ speak-numeral: digits | continuous | inherit ; }
```

Setează rostirea numerelor cifră cu cifră (număr de telefon) sau naturală (sumă de bani).
Opțiuni auditive există, de asemenea, și pentru alte construcții CSS2, cum ar fi tabelele.

Redarea audio a tabelelor

```
{ speak-header: once | always | inherit ; }
```

Setează comportamentul pentru redarea auditivă a tabelului.

Unghiuri

Următoarele unități de măsură descriu unghiuri. Prima este cunoscuta unitate de măsură, gradul, pe care de obicei o indicăm cu un cerc mic, astfel: 90°. Unghiurile sunt măsurate în direcție pozitivă în sensul acelor de ceasornic. Deci `90deg` indică o poziție de la extrema dreaptă, iar `270deg` indică o poziție de la extrema stângă. Unghiurile negative sunt măsurate în aceeași manieră, dar în sens contrar acelor de ceasornic. Deci `-90deg` este echivalent cu `270deg`.

- **`deg` grade sexagesimale** – A 360-a parte a unui cerc. `180deg` se referă la o locație din spatele ascultătorului.
- **`grad` grade centezimale** – Aproape un grad sexazecimal: `360deg = 400grad`.
- **`rad` radiani** – `1deg = π/180rad` sau `360deg = 2πrad`.

Unghiurile pot fi negative și sunt măsurate în sensul acelor de ceasornic, direct din fața ascultătorului. Există anumite calcule matematice în geometria analitică, trigonometrie și în calculul numeric care sunt realizate mai comod atunci când unghiurile sunt măsurate în radiani sau în grade centezimale, dar aceste unități de măsură nu sunt utilizate de obicei decât în aceste contexte matematice și științifice specializate.

Durate

Următoarele unități de măsură descriu durata unui sunet sau a altui eveniment.

- **`ms` milisecunde** – O miime de secundă.
- **`s` secunde** – O secundă obișnuită – a 60-a parte a unui minut.

Duratele nu pot fi negative.

Frecvențe

Următoarele unități descriu frecvențe, numărul de cicluri complete ale sunetului (care generează înălțimea sesizată a sunetului) pe secundă.

- **`Hz` Hertzi** – Numărul de cicluri pe secundă.
- **`kHz` kiloHertzi** – Numărul de cicluri dintr-o miime de secundă, sau o mie de Hertzi.

Frecvențele nu pot fi negative.
Realizarea completă a proprietăților prezentate aici necesită suportul mediilor tridimensionale cu sunet surround; oricum, acestea devin mai obișnuite pe măsură ce jocurile de calculator extrem de realiste cer din ce în ce mai mult de la sistemele gazdă. Chiar și acum sunt disponibile căști cu mai multe surse de sunet care creează o ambianță audio realistă, deci nu va trece mult până când toate aceste lucruri vor fi disponibile la un preț rezonabil.

Dispozitive hand-held

Alt tip obișnuit de afișaj, dispozitivele hand-held, au de obicei ecrane cu rezoluție scăzută, uneori cu o adâncime de culoare de un bit (alb/negru), cu aptitudini de sunet slabe sau inexistente și multe alte limitări. Exemplul perfect de dispozitiv hand-held din zilele noastre este una din diversele variante Palm Pilot, care nu are tastatură și dispune de o grafică alb/negru destul de bună. Recunoașterea scrisului de mână și ecranele tactile înlocuiesc tastatura și mouse-ul. Și, în ciuda faptului că afișajul este foarte mic, puteți accesa o bază de date on-line de pe Web utilizând o conexiune fără fir. Apoi puteți face o investigare, poate chiar o achiziție, toate acestea fără a părăsi șezlongul dumneavoastră confortabil de pe plaja Kona din Hawaii. Figura 13.3 prezintă un netBook bazat pe Psion Java, navigând pe Web și având VGA color pe un dispozitiv hand-held.

FIGURA 13.3
O tastatură și un ecran netBook Psion având o pagină Web afișată de un browser personalizat.

Sistemul de operare nebrevetat EPOC-32 standard, dezvoltat de Psion, a fost adoptat de Ericsson, Nokia și Motorola pentru telefoanele lor mobile cu acces la Web, deci software-ul dezvoltat pentru platforma Psion ar trebui să ruleze fără probleme pe multe platforme. Opera Software din Norvegia are deja o portare inițială a browserului Opera pentru dispozitivele hand-held, cu suport CSS1 complet.

Afișaje miniaturale

Cele mai mici afișaje de până acum sunt cele ale telefoanelor mobile și ale pagerelor alfanumerice, care au de obicei 4 linii a câte 16 caractere (aproximativ) și în timp ce vă uitați la ele nu au alt sonor în afară de niște bipuri. Evident, un telefon are cel puțin și posibilitatea de a livra sunete, dar probabil o singură activitate la un moment dat. În timp ce vă uitați la afișaj nu veți fi capabil să auziți telefonul decât dacă aveți un dispozitiv extern. De asemenea, în timp ce vorbiți la telefon nu puteți vedea afișajul.
Figura 13.4 prezintă telefonul mobil NeoPoint 16000 dual-mode, cu un ecran relativ mare – 11 linii. El este suprapus peste ecranele mai mari ale netBook-ului Psion, full-color VGA, și al seriei Psion 5000, monocrom VGA, de două ori mai mic, pentru a compara aceste dispozitive hand-held mai puternice cu un telefon mobil. Toate aceste ecrane pot utiliza CSS pentru a modifica afișajul în mod dinamic, bazându-se pe caracteristicile efective ale ecranului.

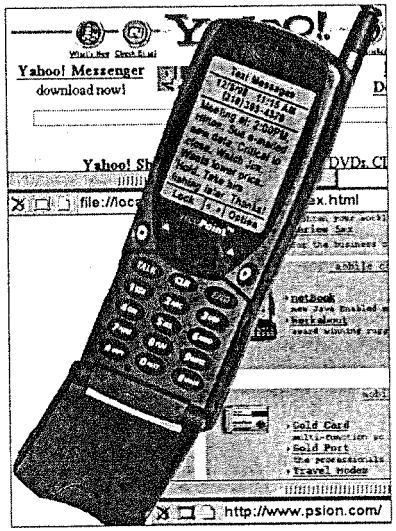


FIGURA 13.4
Telefonul mobil NeoPoint 16000, pregătit pentru Web, este comparat ca dimensiune cu afişajele unor dispozitive hand-held compatibile VGA, cu tastatură completă.

Totuşi, informaţii utile pot fi şi sunt livrate zilnic către astfel de sisteme. De la cotaţii bursiere şi până la mesaje scurte şi notiţe, pagerul a devenit un simbol al epocii noastre moderne. Şi, evident, telefonul mobil este omniprezent, astfel încât oamenii nici nu se mai deranjează să caute un telefon public; întreabă o persoană întâmplătoare dacă este amabilă să îi ofere telefonul său pentru a face un apel urgent.

XML în combinaţie cu XSL/XSLT şi formatări selective de mediu, utilizând fie XSL fie CSS, vor face disponibile mult mai multe informaţii decât aplicaţiile limitate disponibile acum. Pe măsură ce conţinutul de informaţii devine mai accesibil şi mai ieftin de manipulat, cineva ar putea face milioane de dolari cu o aplicaţie care vă permite, de exemplu, să sunaţi de pe telefonul dumneavoastră mobil pentru a afla ora de începere a filmelor şi, după ce aţi ales un spectacol, să descărcaţi pe telefonul dumneavoastră indicaţii pentru traseu şi o hartă. Nu-i aşa că este o nouă lume minunată?

Alte tipuri de mediu

Posibilităţile pentru extinderea tipurilor de mediu sunt nelimitate, de la afişaje tridimensionale şi realitatea virtuală până la adăugarea aptitudinii de a simţi mişcarea unei bărci sau a unei maşini. Acestea sunt deja disponibile prin utilizarea scaunelor care simulează mişcarea şi a ochelarilor care sesizează poziţia din unele sisteme de jocuri pe calculator şi sunt disponibile de multă vreme în mediile militare de vârf şi de simulare de zboruri comerciale şi de vânătoare.

CSS3 va adăuga şi proprietăţi de comportament, făcând posibil ca foile de stil să fie pe deplin interactive şi să încorporeze majoritatea funcţionalităţilor limbajelor de scriptare. Aceasta mai înseamnă că duratele de descărcare şi cerinţele de lăţime de bandă pentru conexiunile lente şi ecranele mici vor scădea drastic, deoarece foile de stil vor fi capabile să îşi dea seama ce tip de afişaj şi de conexiune aveţi şi să îşi alcătuiască cererile către server în conformitate cu acestea.

Depanare

Nimic nu funcţionează aşa cum ar trebui!

După ore de efort, toate în zadar, aţi hotărât că foile de stil nu valorează nici măcar cât un ban găurit.

Este adevărat că foile de stil sunt greu de făcut corect. Cel mai bun sfat este să le puneţi în mâinile unui expert adevărat. Dacă insistaţi să vă luptaţi cu ele, luaţi-o încet. Studiaţi proprietăţile scrise cu aldine din Anexa D şi, pentru început, utilizaţi-le numai pe acestea. În acelaşi timp, ca în exemplul cu citatul evidenţiat pe ecran prezentat anterior, puteţi face efectiv lucruri interesante cu ele, dar nu în toate browserele, dacă utilizaţi acelaşi cod. Acest lucru cere folosirea script-urilor pentru a utiliza selectiv foi de stil diferite pentru fiecare browser. În acest mod vă puteţi concentra pe rând asupra fiecărui browser pentru ca acesta să facă exact ceea ce doriţi dumneavoastră şi să evitaţi bătaia de cap.

De asemenea, puteţi utiliza un editor şi un validator de foi de stil cum este TopStyle, de la `http://www.bradsoft.com/topstyle/`. Este aproape imposibil să ţineţi minte numărul şi identitatea selectorilor şi proprietăţilor suportate de la o versiune la alta a unui browser popular. Deoarece mulţi selectori şi multe proprietăţi din CSS1 au fost etichetate ca „opţionale", mulţi dezvoltatori de software, din neatenţie, au uitat să testeze foile de stil utilizând caracteristicile opţionale şi s-au blocat într-un mod umilitor atunci când le-au întâlnit. Unele browsere nu sunt integrate bine şi au implementat foile de stil utilizând o abordare improvizată, tip tombolă, care oglindeşte abordarea lor cu privire la elemente şi la structura documentului. Aceste browsere sunt înclinate să eşueze atunci când selectorii şi proprietăţile sunt utilizate în combinaţie sau într-un context neprevăzut de autori. Chiar şi cu un sprijin puternic, crearea unei foi de stil care să nu blocheze ceva, pe undeva nu este o operaţie uşoară, chiar dacă sintaxa este perfectă.

Chiar dacă starea actuală a CSS este haotică, noua versiune de la Netscape, după Mozilla, promite un nivel fără precedent al suportului pentru CSS1 şi CSS2 şi, probabil, va urma şi CSS3. Cu această presiune, este posibil ca şi Microsoft să se gândească să se alăture. Deşi browserele mai vechi vor fi întotdeauna cu noi, o abordare server-side simplă poate furniza o pagină minimală, iar browserele moderne pot afişa conţinut bogat în informaţii şi formatat flexibil.

Este adevărat, de asemenea, că suportul CSS este cheia pentru ca Web-ul să devină accesibil pentru toată lumea, de la persoane cu handicap de vedere până la persoane aflate în maşinile lor, care ar fi *mai bine* să nu vadă un ecran video. Cu CSS3 eliminând *orice* nevoie pentru tabelele stil HTML şi având aptitudinea de a schimba foi de stil ale utilizatorului dacă este necesar, proiectantul poate să se concentreze asupra proiectului – partea artistică a operaţiei – şi să ignore detaliile tehnice plictisitoare care ne încetinesc pe toţi.

CAPITOLUL

14

Utilizarea generaţiei următoare de foi de stil extensibile – XSL

În acest capitol

Studiul amănunţit al XSL
Şabloane XSL
Obiecte de formatare XSL
Utilizarea XSL şi XSLT
Depanarea XSL

Studiul amănunţit al XSL

Capitolul anterior compara CSS cu împodobirea unui brad de Crăciun. Ei bine, dacă CSS seamănă cu agăţarea bomboanelor, atunci XSL seamănă cu ingineria genetică. Spre deosebire de CSS, care utilizează documentul în sine ca structură de bază la care să se adauge informaţii de formatare şi de afişare, XSL începe cu citirea arborelui documentului pe care apoi îl reconstruieşte până la conţinutul de bază.

XSL este format din trei părţi principale: XSL propriu-zis, limbajul extensibil al foilor de stil – Extensible Stylesheet Language; XSLT, XSL Transformations (Transformări XSL); şi XPath, limbajul XML Path. XPath nu este deloc XML, ci modul de adresare a fragmentelor de document din interiorul documentelor. XPath, care este studiat amănunţit în Capitolul 8, „XPath", este inclus în mai multe specificaţii, o tendinţă în specificaţiile W3C pe măsură ce ele devin mai modulare şi mai puţin monolitice, deci va fi comod de utilizat în multe contexte. XLink şi XPointer, dezbătute în detaliu în Capitolul 9, „XLink şi XPointer", reprezintă celelalte părţi ale triumviratului de legare al XML.

Atât XSLT cât şi XPath au fost emise ca Recomandări; XSL este singurul aflat încă în stadiul de schiţă de lucru. Unul dintre motive este acela că XSL este foarte dificil. XSL este acolo unde se realizează formatarea, iar grupul de lucru pare să aibă intenţia de a rezolva mai multe lucruri deodată. XSL conţine deja aproape tot din CSS1 şi CSS2, având adăugate într-o măsură semnificativă şi părţi din CSS3. Deoarece CSS3 se află într-un stadiu de dezvoltare preliminar, XSL riscă să se abată de la el dacă specificaţia este fixată prea devreme. Pare foarte probabil să nu fie încheiată înainte de finalizarea CSS3.

Acest capitol cercetează XSL în forma sa actuală, examinând şabloanele XSL, modul de selecţie a elementelor şi fluxul controlului. De asemenea, veţi parcurge mai multe exemple de utilizare a XSL pentru a macheta documentele XML în vederea afişării în browserele moştenite, cel mai des utilizate azi. Aceleaşi tehnici pot fi utilizate pentru a formata şi afişa XML şi în browserele cu suport XML.

Şabloane XSL

Un şablon XSL descrie modul în care elementele sunt comparate şi transformate. Deoarece un şablon poate fi aplicat în mod recursiv, un singur şablon este suficient pentru a se potrivi, dacă este definit corespunzător, unui întreg arbore. Evident, dacă doriţi să efectuaţi lucruri diferite asupra unor părţi diferite ale arborelui documentului dumneavoastră, atunci veţi avea nevoie de mai multe şabloane.

Iată un exemplu de şablon recursiv simplu:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0"
                xmlns:fo="http://www.w3.org/XSL/Format/1.0">
  <xsl:template match="root-document name">
    <!-- aici sunt trecute instructiunile de prelucrare -->
  <xsl:apply-templates select="node()" />
  </xsl:template>
</xsl:stylesheet>
```

Cu aproximaţie, tot ce pare să nu aparţină zonei de nume XSL este trimis către browser. Deci puteţi pune orice şi oricât doriţi în spaţiul indicat, chiar şi o întreagă pagină Web, care, ca prin farmec, va apărea în partea cealaltă. Aceste lucruri sunt într-adevăr şabloane. Tot ce se află în interiorul lor face parte din şablon şi va fi trimis spre ieşire, *excepţie* făcând acele câteva lucruri care au fost etichetate ca şi comenzi XSL prin apariţia în zona de nume XSL. Şi sunt *invizibile,* asemenea acelor tipi îmbrăcaţi în negru din piesele de teatru Noh care grăbesc acţiunea fără ca noi să observăm. Deci orice dintr-o altă zonă de nume, inclusiv zona de nume prestabilită dacă aceasta nu este, din întâmplare, zona de nume XSL, este transferat spre ieşire fără nici o prelucrare ulterioară.

Compararea elementelor

Un element poate fi comparat în mai multe moduri. Cel mai direct mod este prezentat în exemplul anterior; utilizaţi următoarele:

```
<xsl:template match="element-name">
```

Trebuie doar să plasaţi între ghilimele numele efectiv al elementului şi sunteţi gata. Există mai multe moduri de a face acest lucru, unele mai subtile, iar altele nu. Pentru a găsi noduri, XSL utilizează XPath, deci consultarea Capitolului 8 vă poate ajuta în explicarea unor lucruri. Pentru a vă împrospăta memoria, iată o listă cu cei mai obişnuiţi selectori XPath utilizaţi de XSL pentru a identifica locaţii de document cu scopul de a iniţializa trimiterea de cod formatat pentru afişare:

▶ **Vezi** „Utilizarea abrevierilor XPath", la **pagina 174.**

- `paragraph` selectează elementul `paragraph` copil al nodului context.
- `*` selectează toate elementele copil ale nodului context.
- `text()` selectează toate nodurile copil `text` ale nodului context.

- `@name` selectează atributul `name` al nodului context.
- `@*` selectează toate atributele nodului context.
- `paragraph[1]` selectează primul copil `paragraph` al nodului context.
- `paragraph[last()]` selectează ultimul copil `paragraph` al nodului context.
- `*/paragraph` selectează toți nepoții `paragraph` ai nodului context.
- `/doc/chapter[5]/section[2]` selectează cea de-a doua secțiune (`section`) a celui de-al cincilea capitol (`chapter`) din document (`doc`).
- `chapter//paragraph` selectează elementele descendente `paragraph` ale elementului `chapter`, copil al nodului context.
- `//paragraph` selectează toate elementele descendente `paragraph` ale rădăcinii documentului, deci selectează toate elementele `paragraph` aflate în același document cu nodul context.
- `//orderedlist/item` selectează toate elementele `item` aflate în același document cu nodul context care au un părinte `orderedlist`.
- `.` selectează nodul context.
- `.//paragraph` selectează elementele descendente `paragraph` ale nodului context.
- `..` selectează părintele nodului context.
- `../@lang` selectează atributul `lang` al părintelui nodului context.
- `paragraph[@type="warning"]` selectează toți copiii `paragraph` ai nodului context al căror atribut `type` are valoarea `warning`.
- `paragraph[@type="warning"][5]` selectează cel de-al cincilea copil `paragraph` al nodului context care are atributul `type` cu valoarea `warning`.
- `paragraph[5][@type="warning"]` selectează cel de-al cincilea copil `paragraph` al nodului context dacă acel copil are un atribut `type` cu valoarea `warning`.
- `chapter[title="Introduction"]` selectează copilul `chapter` al nodului context care are unul sau mai mulți copii `title` cu valoarea și egală cu `Introduction`.
- `chapter[title]` selectează copilul `chapter` al nodului context care are unul sau mai mulți copii `title`.
- `employee[@secretary and @assistant]` selectează toți copiii `employee` ai nodului context care au atât un atribut `secretary`, cât și unul `assistant`.

În ciuda faptului că puteți ignora majoritatea lucrurilor complexe dacă aveți de gând să faceți doar lucruri simple, pentru a trece într-adevăr prin arbori veți fi probabil nevoit să utilizați sintaxa mai subtilă pentru găsirea de noduri îndepărtate față de locația dumneavoastră prezentă. De fapt, partea cea mai grea a parcurgerii arborelui documentului este urmărirea locului în care vă aflați la un moment dat. Vă este de ajutor dacă aveți un listing la îndemână pentru a putea desena linii numerotate și etichete pe măsură ce vă deplasați brahial de la un punct la altul.

Notă Deplasarea *brahială* este pur și simplu legănarea din creangă în creangă atârnând în mâini, așa cum face o maimuță. Când cineva parcurge un arbore în acest mod, nu trebuie să treacă prin fiecare punct.

Listingul 14.1 prezintă un document XML simplu. În acest capitol veți vedea cum pot afișa browserele moștenite acest document utilizând XSL pentru a genera un document HTML din XML-ul original.

Listingul 14.1 Un model XML de listă de produse

```
<?xml version="1.0"?>
<productList xmlns="http://www.leeanne.com/aristotelian/productlist"
      xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"

xsi:schemaLocation="http://www.leeanne.com/aristotelian/productlist/03xmp01.xsd">
  <!-- Aristotelian Product List -->
  <listTitle>Aristotelian Logical Systems - Product List</listTitle>
  <product>
    <name>European Translator</name>
    <model>Mark IV</model>
    <languages>de fr es en</languages>
  </product>
  <product>
    <name>Universal Translator</name>
    <model>Mark V</model>
    <languages>de fr es en jp</languages>
  </product>
  <product>
    <name>Universal Translator</name>
    <model>Mark VI</model>
    <languages>de fr es en jp ch</languages>
  </product>
  <product>
    <name>BabelFish</name>
    <model>Mark VII</model>
    <languages>de fr es en jp ch kl rm fg</languages>
  </product>
  <product>
    <name>BabelFish</name>
    <model>Mark VIII</model>
    <languages>all {by direct thought transference}</languages>
  </product>
</productList>
```

Observați că acest document conține o listă de produse. Deoarece nu există un motiv întemeiat să presupuneți că numărul de produse este fix, este nevoie de un fel de structură de control care să vă permită să iterați lista. Secțiunea „Prelucrarea și controlul fluxului" din acest capitol descrie amănunțit această structură de control.

Listingul 14.2 conține schema XML pe care o respectă documentul XML.

Listingul 14.2 O schemă XML pentru lista de produse

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xsd:schema PUBLIC "-//W3C//DTD XMLSCHEMA 19991216//EN" "" [
        <!ENTITY % p 'xsd:'>
        <!ENTITY % s ':xsd'>
]>
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema"
            xmlns="http://www.leeanne.com/aristotelian/productlist"
            targetNamespace="http://www.leeanne.com/aristotelian/productlist">
```

Listingul 14.2 (continuare)

```
  <xsd:complexType name="productType" content="elementOnly">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="model" type="xsd:string" />
      <xsd:element name="languages" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="productList">
    <xsd:complexType content="elementOnly">
      <xsd:sequence>
        <xsd:element name="listTitle" type="xsd:string"/>
        <xsd:element name="product" type="productType" minOccurs="1"
➥maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute  name="xmlns:xsi"
                      type="xsd:uriReference"
                      use="default"
                      value="http://www.w3.org/1999/XMLSchema-instance"/>
      <xsd:attribute name="xsi:noNamespaceSchemaLocation"
                      type="xsd:string"/>
      <xsd:attribute name="xsi:schemaLocation"
                      type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Pentru mai multe explicații despre schemele XML, vedeți Capitolul 7, „Scheme XML".

Prelucrarea și controlul fluxului

Pe lângă ramificări utilizând XPath, XSLT are trei, sau poate patru, mecanisme pentru controlul fluxului:

-
```
<xsl:for-each select="node-expression" >
      content
  </xsl:for-each>
```

este instanțiat pentru fiecare nod selectat din lista de noduri curentă.

-
```
<xsl:if test="Boolean test" >
      <xsl:template>
         content
      </xsl:template>
  </xsl:if>
```

este instanțiat de fiecare dată când test este evaluat la true (adevărat). În caz contrar, nu este instanțiat.

-
```
<xsl:choose>
    <xsl:when test="Boolean">
      <xsl:template>
        content
      </xsl:template>
    </xsl:when>
```



```
    <xsl:when test="Boolean">
      <xsl:template>
        content
      </xsl:template>
    </xsl:when>
  </xsl:choose>
```

Primul test xsl:when din listă care este evaluat true, este instanțiat într-un element xsl:choose. Toate celelalte teste xsl:when din elementul xsl:choose sunt ignorate.

-
```
<xsl:otherwise>
    <xsl:template>
      content
    </xsl:template>
  </xsl:otherwise>
```

este instanțiat în cazul în care controlul eșuează la toate testele xsl:when listate anterior într-un element xsl:choose. Elementul xsl:otherwise are sens numai într-o listă xsl:choose. Se poate considera că evaluarea este întotdeauna true și este utilizată astfel:

```
<xsl:choose>
  <xsl:when test="Boolean">
    <xsl:template>
      content
    </xsl:template>
  </xsl:when>
  <xsl:when test="Boolean">
    <xsl:template>
      content
    </xsl:template>
  </xsl:when>
  <xsl:otherwise>
    <xsl:template>
      content
    </xsl:template>
  </xsl:otherwise>
</xsl:choose>
```

-
```
<xsl:sort select="string" lang="??" data-type = "text" | "number" |
  qname-but-not-ncname order = "ascending" | "descending" case-order =
  "upper-first" | "lower-first" />
```

sortează conform criteriului listat, ignorând ordinea naturală a arborelui documentului. Utilizând câteva combinații dintre acestea, ar trebui să fiți capabil să faceți aproape orice cu arborele XML. În primul exemplu de lucru, for-each select="selector" este utilizat pentru a genera un tabel HTML din structura XML utilizând iterații.

Iterații

În XSLT există două metode de iterare. Prima este parcurgerea recursivă a arborelui. Pornind de la rădăcină, puteți vizita recursiv fiecare nod al arborelui, în ciuda faptului că nu aveți control asupra a ceea ce urmează deoarece parcurgeți arborele în ordinea nodurilor. Totuși, și XSL are o construcție XSL:for-each care permite unui șablon să viziteze fiecare element descendent al elementului curent și să îl selecteze pe fiecare. Acest lucru este utilizat pentru a afișa documentul listă de produse, deoarece se pare că

aceasta corespunde cel mai bine la structură.

`<xsl:for-each select="node-expression" >` ... este instanțiat pentru fiecare nod selectat din lista de noduri curentă. Acest lucru vă permite să iterați copiii elementului selectat prin *node-expression* pentru a realiza un fel de prelucrare asupra fiecăruia.

Codul foii de stil prezentat în Listingul 14.3, care acționează asupra documentului definit în Listingul 14.1, generează codul HTML standard din XML-ul original și ar trebui să funcționeze în orice browser care suportă tabele.

Listingul 14.3 O foaie de stil acționând asupra listei de produse XML

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <!-- XSL Stylesheet for Aristotelian Product List -->
  <xsl:template match="/">
    <html>
      <head>
        <title>
          <xsl:value-of select="//productList/listTitle"></xsl:value-of>
        </title>
      </head>
      <body>
        <h2><xsl:value-of select="//productList/listTitle"></xsl:value-of></h2>
        <div>
          <table border="1">
            <tr>
              <th>Product</th>
                <th>Model Number</th>
                  <th>Supported Languages</th>
            </tr>
            <xsl:for-each select="//productList/product">
              <tr>
                <td><xsl:value-of select="name"></xsl:value-of></td>
                  <td><xsl:value-of select="model"></xsl:value-of></td>
                    <td><xsl:value-of select="languages"></xsl:value-of></td>
              </tr>
            </xsl:for-each>
          </table>
        </div>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Figura 14.1 prezintă aspectul documentului în modul de vizualizare browser după transformarea XSL, utilizând XML Spy de la Icon Information Systems, care poate fi găsit la `http://www.xmlspy.com/`.

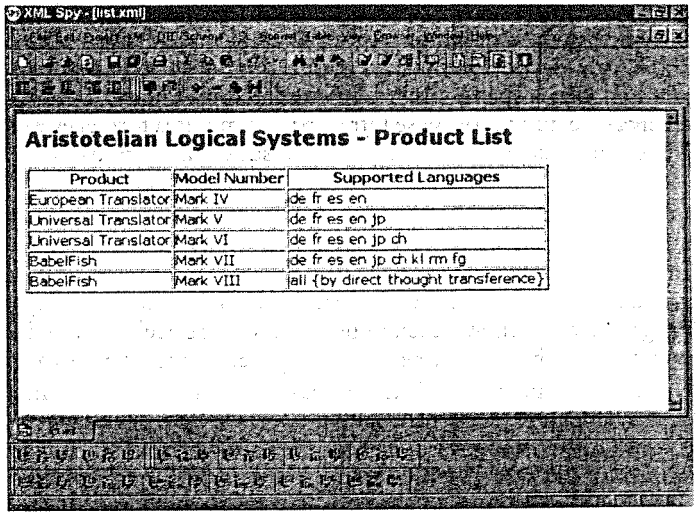


FIGURA 14.1
Lista de produse Aristotelian Logical Systems este prezentată de către XML Spy în modul de vizualizare browser.

Celelalte mecanisme pentru controlul fluxului, secvența standard `if` și secvența `choose...when...otherwise...` utilizată pentru a construi un comutator se comportă într-un mod asemănător și predictibil, deci ele nu sunt tratate aici.

Deși nu este considerată de obicei un mecanism de control al fluxului, puteți utiliza sortarea pentru a modifica ordinea naturală de parcurgere în modul descris anterior în secțiunea „Prelucrarea și controlul fluxului".

Obiecte de formatare XSL

XSL are o listă imensă de *obiecte de formatare* (fo – formatting objects) care seamănă cu proprietățile CSS, deși similaritatea lor nu merge *prea* departe. Asemenea proprietăților CSS, un obiect de formatare XSL are un set de proprietăți care pot fi asociate cu obiectul și care depind de tipul obiectului. Nu există prea multe motoare de redare care să afișeze efectiv obiectele de formatare XSL, deoarece sunt consumatoare de timp și este dificil de implementat o manevrare fără erori a acestor obiecte, iar specificația este un subiect care se poate modifica oricând. Trebuie să fiți foarte hotărât chiar și pentru a le încerca.

InDelv are o implementare parțială, dar este o variantă beta orgolioasă și va rămâne așa până când standardul va deveni stabil. Dar chiar și implementarea celor existente în schița de lucru este cu mult peste posibilitățile oricărui browser existent, deoarece suportul pentru medii paginate conținând limbi cum ar fi chineza – scrisă de sus în jos – este încorporat în specificație, dar nu a mai fost făcut niciodată. W3C plănuiește chiar să adauge obiecte de formatare care nu sunt dreptunghiulare; pe vremuri, acestea ar fi necesitat un meseriaș priceput care-și folosește experiența de o viață pentru a tăia literele astfel încât să arate așa cum trebuie. Dar pe Web, nu știți ce alfabete sunt disponibile pentru a completa nevoia de cuvinte, și nici măcar dimensiunea paginii. Deci fiecare pagină trebuie plănuită și realizată încă o dată.

Orice companie care va încerca acest lucru va avea sigur nevoie de informații despre zeci de limbi, informațiile despre o anumită limbă fiind furnizate de o persoană a cărei limbă

maternă este tocmai limba respectivă, pentru a se asigura că sunt înțelese corect fundamentele limbii. De exemplu, cum aliniați la ambele capete o linie verticală? Sau o linie în arabă, care necesită extinderea formei anumitor litere pentru a ocupa mai mult spațiu, fără a modifica spațiul dintre cuvinte sau dintre litere? Limba arabă utilizează o scriere cursivă care nu are voie să se întrerupă. Frumusețea formei este o parte integrantă a scrierii arabe, iar majoritatea calculatoarelor sunt mai degrabă slabe în aprecierea meritului estetic.

În realitate, tipurile de litere arabe conțin elemente de extindere multiple, care permit unui tipograf să întindă acele caractere atunci când trebuie folosite pentru alinierea la ambele capete prin inserarea uneia sau mai multor linii orizontale de extindere între prima parte a literei (care a fost tăiată în două exact în acest scop) și ultima parte a acesteia. Programele de tehnoredactare pot face în mare același lucru, iar procesoarele XML vor transfera de obicei documentele spre procese externe care vor realiza tehnoredactarea.

Deoarece documentul XML formează, conform definiției, un arbore, ieșirea obiectelor de formatare este de asemenea un arbore. De fapt, este un document XML, în ciuda faptului că este foarte greu de demonstrat acest lucru. XSL folosește zone de nume într-o măsură atât de mare, inclusiv pentru comutatoare interne, încât validarea unui document XSL față de DTD-ul său aproape că depășește puterea calculatoarelor actuale. Anul următor promite să fie unul interesant.

Afișarea unei pagini XML în orice browser

O metodă de machetare a conținutului XML, evident în absența unui XSL adevărat, este să îl transformați în HTML sau în XHTML și să îi aplicați formatări de foi de stil CSS pentru a afișa rezultatul în modul dorit de dumneavoastră. Limitările foilor de stil există încă, deoarece folosiți în continuare acel browser ponosit pe care l-ați utilizat și anul trecut și nu ați obținut prea multe, în afara pașilor suplimentari necesari transformării documentului în XML și transformarea acestuia înapoi din XML. Dacă acestea ar fi fost singurele opțiunile posibile, nici nu v-ați mai fi complicat, însă acum puteți furniza oricărui browser codul corect, utilizând depistări server-side ale browserului.

Suportul XSL, în curs de apariție în Microsoft Internet Explorer și existent deja în browserul InDelv, promite un viitor glorios deoarece XSL și XSLT pot face lucruri la care CSS poate doar visa.

Puteți, de exemplu, să luați un document, să generați automat un cuprins și un index și să le combinați într-o versiune paginată și pregătită pentru tipărire, cât ați clipi din ochi. Ați văzut deja că XSL/XSLT pot fi utilizate pentru a transforma pe loc un document XML într-un document HTML simplu, reutilizând același câmp XML atât pentru câmpul `<title>` cât și pentru antetul inițial `<h2>`. Tehnici asemănătoare pot adăuga, după dorință, obiecte de formatare și transformări mai complexe.

XSL conține mai multe moduri de a schimba machetarea documentelor. Le puteți sorta, puteți elimina ramuri sau adăuga altele noi, adică puteți ajusta arborele documentului până când acesta devine aproape de nerecunoscut. Veți vedea cum se realizează acest lucru spre sfârșitul acestui capitol, în secțiunea „Utilizarea XSL și XSLT".

Manevrarea conținutului XML

Caracterele individuale din datele conținute în elementele nevide sunt de asemenea noduri, deși ar fi obositor să le parcurgeți unul câte unul. Deci XSL furnizează cu amabilitate o scurtătură, selectorul de mulțimi de noduri `node()`. Atunci când este utilizat într-un atribut `select`, ca aici:


```
<xsl:apply-templates select="node()" />
```

procesorul vizitează recursiv fiecare nod copil până când ajunge înapoi. Pe parcurs, el caută automat și transferă textul simplu, găsind de asemenea orice comandă XSL care poate a fost omisă și executând-o în mod corespunzător. Dacă nu veți fi nevoit niciodată să vă răzgândiți, o treabă frumoasă poate fi făcută lăsând „fantomele" `node()` să își facă treaba fără intervenție umană, dacă foaia de stil a fost realizată corect.

Proprietățile de formatare XSL

Proprietățile de formatare XSL sunt aplicate obiectelor de formatare, vizualizări abstracte ale documentului care definesc anumite zone cărora le pot fi aplicate proprietățile. Obiectele de formatare pot fi rafinate prin interacțiunea lor cu documentul, deci obiectul de formatare a corpului unei note de subsol poate să se întindă pe mai multe pagini la afișare, în funcție de criteriile referitoare la suprafața maximă a notelor de subsol, definite pentru document prin intermediul altor obiecte de formatare și proprietăți. De exemplu, corpuri multiple de note de subsol ocupă spațiu pe pagină, deci poate fi necesar ca una sau mai multe note de subsol să fie trecute pe pagina următoare sau chiar să împărțiți o notă de subsol pe mai multe pagini, păstrând totuși o suprafață minimă din textul documentului pe fiecare pagină.

Proprietățile XSL sunt aplicate obiectelor de formatare, deci puteți considera obiectele de formatare ca fiind echivalentul aproximativ al selectorilor CSS, deși proprietatea XSL este, prin proiectare, un echivalent mai precis al unei proprietăți CSS.

Lista proprietăților pentru CSS1 și CSS2 este disponibilă la sfârșitul cărții, în Anexa D „Referințe CSS1, CSS2". Deși există diferențe între aceste valori și aptitudinile „reale" ale XSL, nimeni nu știe de fapt cum va arăta lista la publicarea sa finală ca Recomandare. Totuși, sub rezerva că acestea nu sunt tocmai finale, iată o listă scurtă a obiectelor de formatare XSL și a proprietăților în stil CSS care au fost aprobate de grupul de lucru XSL și se îndreaptă acum mai mult sau mai puțin spre o acceptare mai largă. Să le dorim tot norocul din lume.

Obiecte de formatare pentru paginație și machetare:

```
fo:root
fo:declarations
fo:color-profile
fo:page-sequence
fo:layout-master-set
fo:page-sequence-master
fo:single-page-master-reference
fo:repeatable-page-master-reference
fo:repeatable-page-master-alternatives
fo:conditional-page-master-reference
fo:simple-page-master
fo:region-body
fo:region-before
fo:region-after
fo:region-start
fo:region-end
```

```
fo:flow
fo:static-content
fo:title
```

Obiecte de formatare pentru blocuri:

```
fo:block
fo:block-container
```

Obiecte de formatare in-line:

```
fo:character
fo:initial-property-set
fo:external-graphic
fo:instream-foreign-object
fo:inline
fo:inline-container
fo:leader
fo:page-number
fo:page-number-citation
```

Obiecte de formatare pentru tabele:

```
fo:table-and-caption
fo:table
fo:table-column
fo:table-caption
fo:table-header
fo:table-footer
fo:table-body
fo:table-row
fo:table-cell
```

Obiecte de formatare pentru liste:

```
fo:list-block
fo:list-item
fo:list-item-body
fo:list-item-label
```

Obiecte de formatare pentru legături și obiecte de multiformatare:

```
fo:simple-link
fo:multi-switch
fo:multi-case
fo:multi-toggle
fo:multi-properties
fo:multi-property-set
```

Obiecte de formatare externe:

```
fo:float
fo:footnote
fo:footnote-body
```

Alte obiecte de formatare:

```
fo:wrapper
fo:marker
fo:retrieve-marker
```


Proprietăți de accesibilitate:

```
source-document
role
```

Proprietăți de poziție absolută:

```
absolute-position
top
right
bottom
left
```

Proprietăți auditive:

```
azimuth
cue-after
cue-before
elevation
pause-after
pause-before
pitch
pitch-range
play-during
richness
speak
speak-header
speak-numeral
speak-punctuation
speech-rate
stress
voice-family
volume
```

Proprietăți pentru chenare, margini și fundal:

```
background-attachment
background-color
background-image
background-repeat
background-position-horizontal
background-position-vertical
border-before-color
border-before-style
border-before-width
border-after-color
border-after-style
border-after-width
border-start-color
border-start-style
border-start-width
border-end-color
border-end-style
border-end-width
border-top-color
```

```
border-top-style
border-top-width
border-bottom-color
border-bottom-style
border-bottom-width
border-left-color
border-left-style
border-left-width
border-right-color
border-right-style
border-right-width
padding-before
padding-after
padding-start
padding-end
padding-top
padding-bottom
padding-left
padding-right
```

Proprietăți pentru font:

```
font-family
font-size
font-stretch
font-size-adjust
font-style
font-variant
font-weight
```

Proprietăți pentru despărțirea în silabe:

```
country
language
script
hyphenate
hyphenation-character
hyphenation-push-character-count
hyphenation-remain-character-count
```

Proprietăți pentru margini de bloc:

```
margin-top
margin-bottom
margin-left
margin-right
space-before
space-after
start-indent
end-indent
```

Proprietăți in-line pentru margini:

```
space-end
space-start
```


Proprietăți pentru alinierea zonelor:

```
alignment-adjust
baseline-identifier
baseline-shift
display-align
dominant-baseline
relative-align
```

Proprietăți pentru dimensiunea zonelor:

```
block-progression-dimension
content-height
content-width
height
inline-progression-dimension
max-height
max-width
min-height
min-width
scaling
scaling-method
width
```

Proprietăți referitoare la blocuri și linii:

```
hyphenation-keep
hyphenation-ladder-count
last-line-end-indent
line-height
line-height-shift-adjustment
line-stacking-strategy
linefeed-treatment
space-treatment
text-align
text-align-last
text-indent
white-space-collapse
wrap-option
```

Proprietăți pentru caractere:

```
character
letter-spacing
suppress-at-line-break
text-decoration
text-shadow
text-transform
treat-as-word-space
word-spacing
```

Proprietăți referitoare la culori:

```
color
color-profile-name
rendering-intent
```

Proprietăți referitoare la mobilitate:

```
float
clear
```

Proprietăți de continuitate și întrerupere:

```
break-after
break-before
keep-with-next
keep-with-previous
orphans
widows
```

Proprietăți referitoare la machete:

```
clip
overflow
reference-orientation
relative-position
span
```

Proprietăți pentru linii directoare și linii despărțitoare:

```
leader-alignment
leader-pattern
leader-pattern-width
leader-length
rule-style
rule-thickness
```

Proprietăți pentru legături:

```
active-state
auto-restore
case-name
case-title
destination-placement-offset
external-destination
indicate-destination
internal-destination
show-destination
starting-state
switch-to
```

Proprietăți pentru marcaje:

```
marker-class-name
retrieve-class-name
retrieve-position
retrieve-boundary
```

Proprietăți pentru conversia din număr în șir:

```
format
grouping-separator
grouping-size
letter-value
```


Proprietăți pentru paginare și machetare:

```
blank-or-not-blank
column-count
column-gap
extent
flow-name
force-page-count
initial-page-number
master-name
maximum-repeats
odd-or-even
page-height
page-position
page-width
precedence
region-name
```

Proprietăți pentru tabele:

```
border-collapse
border-separation
caption-side
column-number
column-width
empty-cells
ends-row
number-columns-repeated
number-columns-spanned
number-rows-spanned
starts-row
table-layout
table-omit-footer-at-break
table-omit-header-at-break
```

Proprietăți referitoare la modul de scriere:

```
direction
font-height-override-after
font-height-override-before
glyph-orientation-horizontal
glyph-orientation-vertical
unicode-bidi
writing-mode
```

Alte proprietăți:

```
content-type
id
provisional-label-separation
provisional-distance-between-starts
ref-id
score-spaces
src
visibility
z-index
```

Proprietăți prescurtate:

```
background
background-position
border
border-bottom
border-color
border-left
border-right
border-style
border-spacing
border-top
border-width
cue
font
margin
padding
page-break-after
page-break-before
page-break-inside
pause
position
size
vertical-align
white-space
xml:lang
```

Conținutul generat

Aproape orice puneți în interiorul unui șablon va deveni ieșire atunci când șablonul este prelucrat, atât timp cât nu este un alt element XSL. Deci generarea conținutului este destul de ușoară, după cum ați văzut și în primul nostru exemplu.

În general, construcția următoare este minimul necesar pentru a genera o transformare:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0"
                xmlns:fo="http://www.w3.org/XSL/Format/1.0">
  <xsl:template match="/">
    <!-- aici sunt trecute instructiunile de prelucrare -->
    Fiat Lux!
  <xsl:apply-templates />
  </xsl:template>
</xsl:stylesheet>
```

Aceasta generează o mică transformare la ieșire, care în acest caz este motoul Universității din Berkeley, „Fiat Lux!". Evident, acesta nu este un lucru extraordinar de util de făcut, adică transformarea întregului document în două cuvinte, dar obținerea unei ieșiri efective este distractivă și dovedește că faceți *ceva* corect. De aici încolo puteți să îmbogățiți foaia dumneavoastră de stil, întorcându-vă doar atunci când stricați ceva, iar acel lucru nu mai funcționează.

Elementul `xsl:apply-templates`, din linia șase, prelucrează recursiv toți copiii elementului curent, deci dacă nu sunt necesare prelucrări ulterioare, acest element poate fi eliminat, așa cum s-a procedat în primul exemplu. Prelucrarea ulterioară a fost eliminată deoarece întreaga logică a transformării era conținută în primul șablon. În secțiunea următoare, veți vedea cum să utilizați `xsl:apply-templates` într-o aplicație efectivă.

Sortarea în XSL

Deoarece XSLT acționează asupra vederii arborelui documentului care este păstrată în memorie, structura arborelui poate fi modificată dinamic prin mai multe comenzi. Deci, chiar dacă vă gândiți că va trebui să modificați foarte mult logica programului dumneavoastră pentru a realiza operații asupra unui arbore de document sortat într-o ordine care nu a fost prestabilită, în realitate, tot ce aveți de făcut este să realizați sortarea; toate operațiile din domeniul de valabilitate al sortării vor acționa asupra arborelui ca și când acesta ar fi fost sortat de la început în acest mod. Dacă doriți să refaceți ordinea inițială, atunci va trebui în primul rând să ieșiți din domeniul de valabilitate al procedurii care a sortat lista:

```
<xsl:sort
  select = string-expression
  lang = { nmtoken }
  data-type = { "text" | "number" | qname-but-not-ncname }
  order = { "ascending" | "descending" }
  case-order = { "upper-first" | "lower-first" } />
```

Elementele de sortare sunt utilizate ca și copii ai unui element `apply-templates`. Primul element de sortare identifică cheia primară de sortare, cel de-al doilea identifică a doua cheie de sortare și așa mai departe.

Deci, următorul fragment de foaie de stil va sorta documentul Aristotelian, descris anterior, numai după câmpul nume:

```
<xsl:template match="/">
  <html>
    <head>
      <title>
        <xsl:value-of select="//productList/listTitle"></xsl:value-of>
      </title>
    </head>
    <body>
      <h2>
        <xsl:value-of select="//productList/listTitle"></xsl:value-of>
      </h2>
      <div>
        <table border="1">
          <tr>
            <th>Product</th>
              <th>Model Number</th>
                <th>Supported Languages</th>
          </tr>
          <xsl:apply-templates select="//productList/product">
            <xsl:sort select="name" />
          </xsl:apply-templates>
        </table>
      </div>
    </body>
  </html>
</xsl:template>
```

Apoi, va trebui să definiți un șablon pentru a itera efectiv prin numele sortate și să trimiteți spre ieșire articolele sortate, așa cum s-a procedat în primul exemplu.

Numerotarea în XSL

Numerotarea este o transformare automată, furnizată pentru a vă permite să inserați numere într-un document. Puteți să o utilizați pentru a identifica ordinea nodurilor în document, pentru a insera numere de linie secvențiale, pentru a insera numere de pagină, pentru a identifica numerele capitolelor, pentru a genera un cuprins și altele:

```
<xsl:number
  level = "single" | "multiple" | "any"
  count = pattern
  from = pattern
  value = number-expression
  format = { string }
  lang = { nmtoken }
  letter-value = { "alphabetic" | "traditional" }
  grouping-separator = { char }
  grouping-size = { number } />
```

Modelul XSL Microsoft

Cei de la Microsoft și-au urmat, ca de obicei, propriul drum și în cazul foilor de stil XSL. Ei utilizează un DTD diferit și au presărat standardul cu extensii care nu pot fi utilizate decât în procesoarele Microsoft, deoarece suportă caracteristici specifice Microsoft. În ciuda faptului că Microsoft Internet Explorer este singura versiune de browser important care suportă ceva apropiat de XSL și XSLT, versiunea este suficient de departe de standard, astfel încât majoritatea foilor de stil conforme cu standardul sunt incorecte atunci când sunt citite de MSIE. Viceversa, trecerea într-un alt mediu a unui fișier MSIE care funcționează este de asemenea imposibilă. Microsoft are disponibilă o versiune de tehnologie beta a noului și mult mai compatibilului motor XSL/XSLT pe situl Web al dezvoltatorului, la `http://msdn.microsoft.com/downloads/webtechnology/xml/msxml.asp`, dar ar trebui încorporată în versiunea următoare de Internet Explorer.

Deci, în afară de cazul în care sunteți într-o rețea internă a unei corporații cu reguli draconice referitoare la ce calculatoare pot fi instalate și cu ce software – toate Intel și Microsoft – nu prea aveți noroc, iar XML/XSL este doar un alt dialect Microsoft pentru care sunteți nevoit să scrieți rutine speciale. Asta dacă nu cumva preluați concepția Microsoft și insistați că restul lumii pur și simplu nu vede avantajele de a fi în ton cu tot ceea ce iese din Redmond.

Nutrim speranța că, așteptata apariție a produsului Mozilla, ca generația următoare a browserului Netscape, va pune puțin în dificultate Microsoft pentru a se conforma, dar nu aș băga mâna în foc. Dialectele sistemelor de operare servesc companiilor cam în același mod în care servește argoul copiilor – îi face să pară la modă în ochii prietenilor, dar pe adulți îi face să pară ridicoli atunci când încearcă să le vorbească. Dar acest lucru lasă puțin de dorit atunci când oamenii au ceva serios de făcut.

Până acum, majoritatea siturilor Web au fost orientate spre piață, iar partizanii acestei orientări vor face orice pentru ca mesajul lor să aibă succes, inclusiv realizarea de versiuni diferite ale siturilor lor pentru a obține un efect maxim pentru fiecare browser. Campaniile de marketing țipătoare cum sunt cele de la Coca Cola și altele de același gen nu se vor opri în fața unor obstacole de comunicare lăsate de producătorii de browsere. Vor angaja consultanți, vor aduna niște grupuri focus și vor face ca situl lor să *arate* ca și când ar fi fost proiectat de ultimul idol al muzicii, chiar dacă proprietarii poartă costume Brooks Bros., cămăși scrobite și cravate sobre.

Dar persoanele care nu au nici o legătură cu marketingul au lucruri mai importante de făcut decât să se joace cu dialectele browserelor, după cum arată imensa cerere de XML. Aceleași persoane sunt relativ intolerante față de orice lucru care fie costă bani, fie stă în calea câștigului de bani, lucru făcut de altfel și de variatele standarde de browsere. Studiile au arătat că luarea în considerare a diferențelor brevetate în tehnologia browserelor adaugă între 25 și 50% la timpul necesar proiectării și la costul unui proiect Web mediu. Dialectele browserelor vor converge într-un mod sau altul, deoarece un impact negativ asupra rezultatelor finale nu este privit cu ochi buni în cele mai înalte cercuri financiare.

Browserul Opera

Dacă vreți să aveți o idee despre ce poate face un browser bazat pe standarde, aruncați o privire asupra versiunii beta a browserului Opera 4.0 de la Opera Software, la `http://www.operasoftware.com/`, și vedeți cât de ușor poate face anumite lucruri. V-ați jucat vreodată cu copierea de imagini de pe un sit? (Nu că am pleda în favoarea încălcării drepturilor de autor ale cuiva!) Încercați să utilizați Opera, care are în meniu o opțiune „Copy File and Images" (copiere fișiere și imagini).

Ați obosit încercând să descifrați litere mici de culoare albastru-închis pe o pagină neagră cu modele ondulate care vă dau dureri de cap? Opera are la îndemână un mic buton, în partea de jos a ecranului, care comută între opțiunile de stil ale sitului Web și opțiunile de afișare ale utilizatorului, permițându-vă să reformatați instantaneu neplăcuta pagină la litere negre, mari, pe o pagină albă, imaculată. Mai există un buton cu care puteți renunța la încărcarea imaginilor în zilele în care rețeaua pare să se târască, economisind o mulțime de timp.

Versiunea 4.0 a browserului Opera, acum în variantă beta, adaugă suport pentru XML și CSS1/2 și, în demonstrațiile de până acum, arată promițător, deși nu va avea suport complet pentru XSL până la lansarea Recomandării XSL a W3C. Implementarea foilor de stil CSS1/2 în Opera este concurată numai de Mozilla și Emacs/W3, iar browserul este mic și ieftin. Totul încape perfect pe o dischetă. Vi le amintiți? Înainte ca aplicații de gigaocteți să vă umple hard-discul?

Utilizarea XSL și XSLT

Ca exemplu, priviți o înregistrare XML reală dintr-o bază de date cu recenzii de cărți. Listingul 14.4 prezintă schema XML, iar Listingul 14.5 prezintă documentul XML.

Listingul 14.4 Un DTD XML pentru recenzii

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- document root -->
<!ELEMENT review (page-header, review-body)>
<!-- elements in alphabetical order -->
<!ELEMENT blurb (blurb-text | web-address | blurb-middle)*>
<!ELEMENT blurb-middle (#PCDATA)>
<!ELEMENT blurb-text (#PCDATA)>
<!ELEMENT body (book-title, book-author, book-publisher, byline, review-text)>
<!ELEMENT book-author (#PCDATA)>
```

Listingul 14.4 (continuare)

```
<!ELEMENT book-copyright (book-text, book-title, book-copyright-text)>
<!ELEMENT book-copyright-text (#PCDATA)>
<!ELEMENT book-publisher (web-address, publish-date, publish-location)>
<!ELEMENT book-text (#PCDATA)>
<!ELEMENT book-title (#PCDATA)>
<!ELEMENT byline (byline-text, mail-address)>
<!ELEMENT byline-text (#PCDATA)>
<!ELEMENT citation (#PCDATA)>
<!ELEMENT commentary (plaintext | citation)*>
<!ELEMENT copyright-reserved (#PCDATA)>
<!ELEMENT footer (review-copyright, book-copyright, blurb)>
<!ELEMENT header (prefix, short-title)>
<!ELEMENT host-name (#PCDATA)>
<!ELEMENT mail-address (mail-locator)>
<!ATTLIST mail-address
          refer CDATA #REQUIRED
>
<!ELEMENT mail-locator (#PCDATA)>
<!ELEMENT page-header (page-title)>
<!ELEMENT page-title (#PCDATA)>
<!ELEMENT plaintext (#PCDATA)>
<!ELEMENT prefix (#PCDATA)>
<!ELEMENT publish-date (#PCDATA)>
<!ELEMENT publish-location (#PCDATA)>
<!ELEMENT publisher-name (#PCDATA)>
<!ELEMENT review-body (header, body, rule, footer)>
<!ELEMENT review-copyright (review-text, mail-address, copyright-reserved)>
<!ELEMENT review-text (#PCDATA | commentary)*>
<!ELEMENT rule EMPTY>
<!ELEMENT service-name (#PCDATA)>
<!ELEMENT short-title (#PCDATA)>
<!ELEMENT web-address (web-locator)>
<!ATTLIST web-address
          refer CDATA #REQUIRED
>
<!ELEMENT web-locator (publisher-name | service-name | host-name)>
```

Listingul 14.5 Un document XML pentru recenzii

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE review SYSTEM "http://www.leeanne.com/taborsarah/13xmp03.dtd">
<review>
 <page-header>
  <page-title>Review: A Ritual of Drowning</page-title>
 </page-header>
 <review-body>
  <header><prefix>Review: </prefix><short-title>A Ritual of Drowning </short-title>
  </header>
  <body>
   <book-title>A Ritual of Drowning - Poems of Love and Mourning</book-title>
   <book-author>Teya Schaffer</book-author>
```



```
  <book-publisher>
   <web-address refer="http://www.leeanne.com/taborsarah/index.html">
    <web-locator><publisher-name>Tabor Sarah Books</publisher-name></web-locator>
   </web-address>
   <publish-date>1999</publish-date>
   <publish-location>Oakland and Palo Alto</publish-location>
  </book-publisher>
  <byline>
   <byline-text>Reviewed by Lee Anne Phillips </byline-text>
   <mail-address refer="mailto:Lee Anne Phillips &lt;leeanne@leeanne.com&gt;">
    <mail-locator>&lt;leeanne@leeanne.com&gt;</mail-locator>
   </mail-address>
  </byline>
  <review-text>
  <commentary><plaintext>
   How many ways can a heart break and yet be mended? In this
   heartrending but essential collection by Teya Schaffer,
   grief is examined in raw, searing detail and transformed
   into the highest art. I was reminded of Jane Yolen's <citation>
   Cards of Grief</plaintext></citation><plaintext> by the power and beauty of
➥the whole
   and reduced to helpless tears by many of the individual
   parts. Is this a lamentation? A love story? Or some of
   both? Or perhaps even a paean of praise. It starts,
   thankfully, with a strong statement of reconciliation that
   helps us through the rough spots to come as we prepare to
   follow Teya through her lover's final illness and death.</plaintext>
   </commentary>
  </review-text>
 </body>
 <rule/>
 <footer>
  <review-copyright>
   <review-text>Review Copyright &#169; 1999, by Lee Anne Phillips
➥</review-text>
   <mail-address
   refer="mailto:Lee%20Anne%20Phillips%20&lt;leeanne@leeanne.com&gt;">
    <mail-locator>&lt;leeanne@leeanne.com&gt;</mail-locator>
   </mail-address>
   <copyright-reserved> - All Rights Reserved Worldwide</copyright-reserved>
  </review-copyright>
  <book-copyright>
   <book-text>Poem excerpts are from </book-text>
    <book-title>A Ritual of Drowning - Poems of Love and Mourning</book-title>
    <book-copyright-text>
     Copyright &#169; 1999 by Teya Schaffer - All Rights Reserved Worldwide
    </book-copyright-text>
  </book-copyright>
  <blurb>
   <blurb-text>For more reviews of outstanding books by
              women, please visit the </blurb-text>
    <web-address refer="http://www.cybergrrl.com/review/">
     <web-locator><service-name>Women's Books Online</service-name></web-locator>
```

Listingul 14.5 (continuare)

```
    </web-address>
    <blurb-middle>pages kindly hosted by</blurb-middle>
    <web-address refer="http://www.cybergrrl.com/">
     <web-locator><host-name>Cybergrrl!</host-name></web-locator>
    </web-address>
   </blurb>
  </footer>
 </review-body>
</review>
```

Aici există o singură înregistrare, dar ar trebui să fie suficientă. În înregistrare nu există nimic care să poată fi ignorat, deoarece întreaga recenzie ar trebui publicată, iar dumneavoastră ați adăugat deja o instrucțiune de prelucrare care indică la foaia de stil încă inexistentă. Mai există câteva lucruri de adăugat între anumite câmpuri, dar puteți ajunge la ele atunci când începeți să cercetați câmpurile individuale.

Inițializarea arborelui de transformare

Pentru început, trebuie să inițializați arborele dumneavoastră de transformare. Deoarece ceea ce faceți poate începe cu orice document, începeți folosind ca selector o abreviere generală de rădăcină (/):

```
<xsl:template match="/">
  <!-- aici intra instructiuni de prelucrare -->
</xsl:template>
```

Această instrucțiune selectează nodul rădăcină în orice document. Toate celelalte noduri sunt copiii acestui nod, deci vă aflați în locul potrivit. Acum trebuie să știți ce să faceți cu informațiile din înregistrare. În acest moment le puteți transforma în HTML, deși acest lucru ridică o întrebare; oricum, HTML este suportat de marea majoritate a browserelor moștenite, deci de ce să vă luptați cu destinul? Browserele moștenite nu vor dispărea curând, deci majoritatea paginilor XML vor fi servite multor utilizatori în forma HTML.

De asemenea, ar fi frumos să trimitem spre ieșire și instrucțiunile de formatare, deoarece acestea nu funcționează deocamdată prea bine. În cele din urmă, un browser cu suport XML va ști cum să afișeze obiectele de formatare în loc să afișeze HTML fad, dar aceasta mai târziu. Aceleași tehnici pot fi folosite pentru a realiza oricare operație. Din moment ce vă aflați aici, introduceți elementul de împachetare `xsl:stylesheet` și declarațiile de zonă de nume:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0"
                xmlns:fo="http://www.w3.org/XSL/Format/1.0">
  <xsl:template match="/">
    <!-- aici intra instructiuni de prelucrare -->
  </xsl:template>
</xsl:stylesheet>
```

După o editare considerabilă, veți încheia cu codul prezentat în Listingul 14.6.

Listingul 14.6 O foaie de stil XSL pentru documentul de recenzii

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform">
  <!-- XSL Stylesheet for Aristotelian Product List -->
  <xsl:template match="/">
    <html>
      <head>
        <title>
          <xsl:value-of select="//review/page-header/page-title" />
        </title>
      </head>
      <body>
        <h2>
          <xsl:value-of select="//review/page-header/page-title" />
        </h2>
        <blockquote>
          <h3><xsl:value-of
            select="//review/review-body/body/book-title" /></h3>
          <blockquote>
            <h3>by <xsl:value-of
              select="//review/review-body/body/book-author" /></h3>
            <xsl:for-each
              select="//review/review-body/body/book-publisher/web-address">
              <h3><a><xsl:attribute name="href"><xsl:value-of
              select="refer" /></xsl:attribute><xsl:value-of
              select="web-locator/publisher-name" /></a>,
              <xsl:value-of
                select="//review/review-body/body/book-publisher/publish-date" />;
              <xsl:value-of
                select="//review/review-body/body/book-publisher/publish-
➥location" /></h3>
            </xsl:for-each>
          </blockquote>
        </blockquote>
        <blockquote>
          <h3><xsl:value-of
            select="//review/review-body/body/byline/byline-text" />
          <xsl:for-each
            select="//review/review-body/body/byline/mail-address">
            <a><xsl:attribute name="href"><xsl:value-of
                select="refer" /></xsl:attribute><xsl:value-of
                select="mail-locator" /></a>
          </xsl:for-each>
          </h3>
        </blockquote>
        <xsl:for-each select="//review/review-body/body/review-text/commentary">
          <blockquote>
            <xsl:for-each select="*">
              <xsl:if xsl:test="contains(string(local-name()),'plaintext')">
                <xsl:value-of select="." />
              </xsl:if>
              <xsl:if xsl:test="contains(string(local-name()),'citation')">
                <cite><xsl:value-of select="." /></cite>
```

Listingul 14.6 (continuare)

```
            </xsl:if>
          </xsl:for-each>
        </blockquote>
      </xsl:for-each>
      <xsl:for-each select="//review/review-body/rule">
        <blockquote>
          <hr />
        </blockquote>
      </xsl:for-each>
      <blockquote>
        <xsl:value-of select="//review/review-body/footer/review-
➥copyright/review-text" />
        <xsl:for-each
          select="//review/review-body/footer/review-copyright/mail-address">
          <a><xsl:attribute name="href"><xsl:value-of
              select="refer" /></xsl:attribute><xsl:value-of
              select="mail-locator" /></a><xsl:value-of
              select="//review/review-body/footer/review-copyright/copyright-
➥reserved" />
        </xsl:for-each>
      </blockquote>
        <br />
      <blockquote>
        <xsl:for-each
          select="//review/review-body/footer/book-copyright">
              <xsl:value-of select="book-text" />
              <cite><xsl:value-of select="book-title" /></cite>
              <xsl:value-of select="book-copyright-text" />
        </xsl:for-each>
      </blockquote>
        <br />
      <blockquote>
        <xsl:for-each
          select="//review/review-body/footer/blurb">
              <xsl:value-of select="blurb-text" />
          <xsl:for-each
            select="web-address"><a><xsl:attribute
                name="href"><xsl:value-of
                select="refer" /></xsl:attribute><xsl:value-of
                select="web-locator/service-name" /></a>
          </xsl:for-each>
          <xsl:value-of select="blurb-middle" />
          <xsl:for-each
            select="web-address[1]"><a><xsl:attribute
                name="ref"><xsl:value-of
                select="refer" /></xsl:attribute><xsl:value-of
                select="web-locator/host-name" /></a>
          </xsl:for-each>
        </xsl:for-each>
      </blockquote>
    </body>
  </html>
```



```
  </xsl:template>
</xsl:stylesheet>
```

Depanarea XSL

Nici foile de stil XSL nu funcționează!

Ați abandonat CSS în favoarea noilor și lejerelor XSL și XSLT. Spre marea dumneavoastră surpriză, nici XSL nu funcționează.

Ei bine, deocamdată este prea devreme. În momentul scrierii acestei cărți, browserul Microsoft Internet Explorer este singura versiune de browser comercial pregătită cu adevărat să suporte XML și nu este chiar conformă cu standardul în starea sa actuală. Zeci de pachete de creare de documente suportă schița de standard XSL, cel puțin în variantă beta, cum sunt mediile de editare XML Spy și InDelv XML, precum și SAXON, Jade și altele. Jucați-vă câteodată cu ele, iar până la urmă totul va ieși bine.

Partea a III-a

Integrarea XML cu alte tehnologii

15 Obținerea informațiilor despre utilizator
16 Securitate și confidențialitate
17 Utilizarea facilităților server-side – Java
18 Utilizarea facilităților server-side – ActiveX
19 Utilizarea brokerilor de obiecte comune
20 Planificarea resurselor pentru întreprinderi în XML

Obţinerea informaţiilor despre utilizator

CAPITOLUL 15

În acest capitol

- Profiluri de utilizator
- Negocierea profilurilor
- Utilizarea cookie-urilor pentru întreţinerea unui profil
- Utilizarea unei baze de date server-side
- ID-uri şi parole utilizator
- P3P – Platforma pentru preferinţe de confidenţialitate
- OPS/P3P – Standardul deschis pentru profiluri (Open Profiling Standard)
- JavaScript şi limbaje de scriptare
- Studii de caz

Profiluri de utilizator

Pe măsura dezvoltării Web-ului, informaţiile adunate de situri despre utilizatori au explodat în nume detaliate, adrese, adrese de e-mail, informaţii despre cartea de credit, obiceiuri de cumpărător, date demografice complete sau chiar rasiale (informaţii despre culoarea feţei şi a părului care ar permite stabilirea identităţii rasiale).

În Statele Unite, sunt puţine restricţii privind informaţiile care pot fi colectate, iar limita datelor care pot fi vândute altora este inexistentă. S-a observat că există mai multe restricţii la divulgarea numelor filmelor închiriate de dumneavoastră de la un magazin de închiriere de casete decât la datele de natură intimă, medicale sau financiare, de pe Web. În acelaşi timp, această informaţie este şi un obiect de uz valoros. Numele şi datele pot fi vândute aproape oricui, iar advertiserii[1] doresc, chiar sunt nerăbdători, să le cumpere.

Colectarea datelor este tolerată deoarece, în cele mai multe cazuri, oamenii nu ştiu că datele sunt vândute unei terţe părţi. Guvernul S.U.A. nu prea este interesat de protejarea acestor date şi permite să se facă aproape orice cu ele. Nici măcar nu are de gând să propună o legislaţie pentru utilizatori care să controleze folosirea adreselor lor de e-mail, aprobând tacit bombardarea cetăţenilor americani cu spam-uri[2].

Dar *unii* oameni *s-au* alarmat. Multe ţări au restricţii rigide cu privire la informaţiile personale care trec de graniţele naţionale, chiar dacă în interiorul ţării respective sunt oferite gratuit spre utilizare. Uniunea Europeană a emis directive de confidenţialitate care controlează ce pot face siturile Web cu informaţiile colectate. Deci, toţi operatorii de situri Web care au buzunare suficient de adânci trebuie să aibă stabilită o anumită politică comprehensivă de confidenţialitate, în caz contrar putând fi daţi în judecată în Germania pentru destăinuirea de informaţii dintr-o tranzacţie iniţiată de un cetăţean german. Cea mai sigură politică pentru corporaţii cu prezenţă internaţională este probabil aceea să nu divulge niciodată date personale identificabile unei terţe părţi.

Pe acest fundal, Consorţiul World Wide Web (W3C) şi-a început lucrul la un limbaj bazat pe XML, numit de ei Platforma pentru preferinţe de confidenţialitate (P3P – Platform for Privacy Preferences). W3C este format în principal din persoane care au interese economice importante în păstrarea în cea mai mare parte a stadiului actual, deci P3P este un compromis între a permite utilizatorilor să îşi controleze informaţiile proprii şi a permite siturilor să pretindă informaţii personale înainte de a-l servi pe utilizator.

[1] N.T. – Persoane sau firme care se ocupă de publicitate şi comerţ.

[2] N.T. – Mesaje publicitare sau de altă natură care nu sunt solicitate de destinatar.

Negocierea profilurilor

În primul rând, multe situri nu vor utiliza P3P. Dacă un sit popular nu utilizează această platformă, atunci vor exista presiuni importante asupra utilizatorilor să renunţe la confidenţialitate de dragul accesării sitului. Încercaţi numai să navigaţi prin situl Web al Microsoft având dezactivate cookie-urile, pentru a primi o mostră despre dificultatea impunerii dreptului dumneavoastră la confidenţialitate.

Apoi, dacă accesaţi situl TV Guide, la `http://www.tvguide.com/`, şi le permiteţi să colecteze informaţiile de identificare şi codul ZIP al companiei dumneavoastră de cablu, ei vor oferi în schimb o planificare localizată a programelor TV disponibile în localitatea dumneavoastră. Ei stochează informaţia într-un cookie, deci nu va trebui să o introduceţi din nou. Intruziunea este minimă, deşi se poate presupune că situl ar putea vinde datele colectate, date brute despre utilizatorii Internet conţinând codul ZIP şi informaţii despre serviciul de televiziune prin cablu la care sunt abonaţi. În acelaşi mod, pentru informaţii meteo locale, cumpărături on-line comode şi asistenţă în cazul problemelor cu calculatorul se pot cere date de la utilizator înainte de a face disponibile acele servicii.

Din experienţa cu ofertele actuale de la diverşi furnizori, orice poate fi oferit ca stimulent, de la bani şi până la acces gratuit la servicii care în mod normal ar trebui plătite, pentru a afla cât mai multe date personale. De fapt, această informaţie este atât de valoroasă încât unele companii sunt în stare să vă ofere gratuit un calculator şi acces la Internet pentru a-i lăsa pe advertiseri să vă inunde ecranul cu reclame din domeniile dumneavoastră de interes.

Sugestie de la LeeAnne.com Words to weave by

Un exemplu de troc între confidenţialitate şi beneficii este PrivaSeek (`http://www.privaseek.com/`), un broker on-line de informaţii despre utilizatori. După ce vă abonaţi la acest serviciu şi dezvăluiţi informaţii personale, serviciul se oferă să vă protejeze confidenţialitatea numai dacă dumneavoastră nu doriţi să o scoateţi la lumină. Dacă renunţaţi la confidenţialitate, puteţi primi o mică recompensă de la advertiseri sau de la proprietarii de situri, iar PrivaSeek obţine un comision.

Din păcate, după ce aţi transferat datele dumneavoastră unei companii, aceasta poate să facă aproape tot ce vrea cu aceste date fără a vă anunţa, iar în Statele Unite aproape fără repercusiuni. Deoarece există puţine legi referitoare la vânzarea de date personale în

Statele Unite, singurul mod prin care vă puteți apăra drepturile împotriva unei companii ar fi intentarea unui proces civil pentru încălcarea contractului sau fraudă, dar P3P nu este proiectat pentru schimbul sau garantarea informațiilor în mod contractual. De fapt, schimbul de informații în P3P este indiscutabil unilateral. Există multe câmpuri care facilitează schimbul de date personale de la un utilizator la un sit care cere aceste date, deși conținutul exact al acelor câmpuri nu este dezvăluit cu claritate. Spre deosebire de aceasta, există extrem de puține exemple de cereri reciproce, în care un utilizator să obțină informații despre o companie.

Deocamdată, singurul câmp de date de identificare „obligatoriu" de completat pentru situl Web în sine este atributul entitate al elementului de politică de date, care *ar trebui* să conțină numele entității și informațiile fizice de contact. Dar nu există un mecanism real de constrângere, iar câmpul nu este structurat, deci orice apare în câmp este lăsat în seama imaginației proiectantului. Browserul nu va fi capabil să vă spună dacă lipsește ceva, deoarece câmpul nu este structurat. Singurul mod prin care un utilizator poate să se convingă că în câmpul entitate se află într-adevăr informații de contact utile este să îl studieze înainte de a continua. Pare mai degrabă o improvizație târzie ca răspuns la o critică amară decât o caracteristică bine gândită.

Un sit Web poate cere adresa și numărul de telefon al unui utilizator folosind protocolul P3P, dar nu există nici o modalitate ca un utilizator să ceară adresa fizică și informațiile de contact prin telefon ale unui sit Web înainte de realizarea unei tranzacții, decât dacă citește informația entitate înainte de a continua. A sugera că nu ar trebui să faceți afaceri cu un sit care nu are suficientă încredere în dumneavoastră pentru a vă spune unde se află pare inutil, dar poate v-ar surprinde dacă ați ști cât de multe situri Web fac această informație extrem de dificil de obținut sau chiar imposibil.

Iar situația jenantă a mecanismului de contact entitate ridică următoarea întrebare: Care este rostul unui sistem automatizat dacă necesită intervenția umană pentru a începe fiecare tranzacție? Și cum se pot salva informațiile de contact pentru o utilizare ulterioară fără a le scrie undeva?

Utilizarea cookie-urilor pentru întreținerea unui profil

Spre deosebire de modelul P3P bazat pe XML, care este transparent și interactiv, utilizarea de cookie-uri este un răspuns brutal la problema controlului confidențialității, deoarece acestea sunt stocate pe calculatorul utilizatorului și pot fi șterse ușor de acesta dacă dorește. Totuși, ele nu sunt transparente și uitându-vă la ele nu puteți spune de obicei ce conțin. Deci trebuie să păstrați sau să ștergeți cookie-urile după principii generale, și nu printr-o alegere rațională, bazată pe datele conținute de acestea. De asemenea, situl care setează cookie-urile are deja control asupra datelor dumneavoastră, iar stocarea informației pe sit sau vânzarea ei este decizia lor, și nu a dumneavoastră. De fapt, în Statele Unite este perfect legal transferul informațiilor din cookie-uri unei alte firme, deci mai mulți furnizori pot utiliza o singură bază de date server-side despre clienți, prin transferarea de la unul la altul a unui index din baza de date, care poate fi stocat într-un cookie. Seamănă puțin cu tatuarea unui număr de identificare pe brațul dumneavoastră sub forma unui cod de bare, astfel încât comercianții să știe chiar în momentul în care ați intrat în magazinul lor virtual cine sunteți și câți bani aveți în portofel.

Chiar și pe siturile care susțin că protejează confidențialitatea dumneavoastră on-line, advertiserii care plasează acele omniprezente bannere nu fac parte din punct de vedere tehnic din sit, deci nu sunt constrânși să respecte o astfel de politică. De fapt, ei alcătuiesc și vând mari baze de date cu informații personale despre obiceiurile de navigare ale indivizilor. Deci chiar și siturile care afirmă că nu adună *nici* o informație despre dumneavoastră pot cumpăra și utiliza aceleași informații de la o terță parte, sau chiar de la o filială, și apoi să le revândă fără nici un impediment.

Cookie-urile nu sunt portabile între browsere. Deci dacă utilizați mai multe browsere, atunci va trebui să reintroduceți datele pentru fiecare, pierzând puțin din comoditatea promisă. De asemenea, dacă ștergeți periodic cookie-urile, informația conținută în ele va trebui introdusă din nou atunci când va apărea următoarea interogare a sitului care a cerut-o de la început.

Ștergerea cookie-urilor dumneavoastră poate reprezenta o sursă de frustrări și pentru proprietarii sitului, deoarece informațiile lor despre cumpărăturile făcute și cele demografice, colectate cu multă grijă, pot fi șterse prin câteva acționări de taste ale utilizatorului, permițând clienților vechi să se reîntoarcă într-o clipită la o înregistrare curată. Aici ar trebui să ne oprim și să îi lăsăm să ne plângă pe umăr.

Browserul Opera, de la `http://www.opera.com`, are mai multe caracteristici proiectate să vă ajute la protejarea confidențialității dumneavoastră on-line, printre care și abilitatea de a șterge automat toate cookie-urile noi atunci când închideți browserul, precum și posibilitatea de a preveni ca limbajele de scriptare să fie executate sau ca ferestre noi să apară automat pe ecranul dumneavoastră. Ca o compensare a acestui lucru, el are un cache intern cu date personale care poate fi utilizat pentru a completa repede formulare atunci când veți dori acest lucru.

Cookie-urile sunt și dependente de sit, deci fiecare sit care depozitează un cookie în calculatorul dumneavoastră se stochează pe sine. Dacă navigați intens, atunci pe calculatorul dumneavoastră se pot aduna mii sau chiar zeci de mii de cookie-uri, transferând către dumneavoastră întregul cost al menținerii bazei lor de date. Deoarece majoritatea informațiilor conținute în aceste mii de intrări sunt copiate de la sit la sit și era necesar să fie introduse iar și iar, P3P promite măcar să ne scape de introduceri plictisitoare și repetitive de date, chiar dacă beneficiile sale de confidențialitate sunt minime.

Utilizarea unei baze de date server-side

Dacă nu există un suport P3P nativ, alternativa la cookie-uri este stocarea informațiilor într-o bază de date pe serverul sitului sau la un broker intermediar, cum ar fi PrivaSeek, TRUSTe, BBB Online sau PrivacyBank.com. Deși în acest mod se permite ca informația să fie stocată fără costuri pentru utilizator, apare problema modului în care această informație poate fi vizualizată și gestionată de către acesta. Sau chiar șterasă, dacă utilizatorul dorește să revină la o confidențialitate deplină după ce a permis un acces limitat la datele sale personale.

Actualmente, în Statele Unite nu există metode de a recupera datele după ce cineva a obținut acces la acestea, indiferent de înțelegerea făcută cu persoana care le-a primit inițial. Deci, deși RealNetworks susține că vă garantează confidențialitatea, această înțelegere nu se extinde și la RealJukebox, care rulează pe calculatorul dumneavoastră și nu este un sit Web. Deci RealNetworks poate să vândă preferințele dumneavoastră mu-

zicale la terțe părți fără știrea sau consimțământul dumneavoastră, și chiar o face. TRUSTe este de părere că aceasta nu contravine politicilor lor, în ciuda faptului că utilizatorii obișnuiți nu vor înțelege cum și de ce sunt permise excepții la politicile de confidențialitate stabilite.

Atenție Cel puțin câțiva dintre așa-zișii garanți de confidențialitate nu au adresă fizică sau număr de telefon disponibile pe propriul sit Web. În acest mod, utilizatorul este pus în postura incomodă de a se încrede într-o companie fără să știe unde să o găsească dacă apare o dispută. Acest lucru ar trebui să vă facă atent atunci când alegeți unul dintre numeroasele servicii disponibile.

Sugestie de la LeeAnne.com Words to weave by

Brokerii de servicii de confidențialitate garantează, teoretic, o modalitate prin care utilizatorul poate să editeze sau chiar să șteargă informațiile proprii. Este de presupus că aceștia vor impune politici de refolosire, la fel ca furnizorii de liste poștale, pentru a se asigura că un advertiser nu poate să obțină datele o singură dată iar apoi să își mențină propria bază de date. Dar nu există nici o garanție că se va întâmpla așa și, de fapt, este destul de ușor pentru un sit să se descotorosească de înțelegerile sale prin mijloace care sunt complet invizibile pentru utilizator. Se pot face atât de mulți bani prin vânzarea informațiilor dumneavoastră personale încât va exista întotdeauna tentația de a înșela și a induce în eroare, deoarece nu prea există soluții legale care să vă protejeze confidențialitatea. Singura soluție reală este o pedeapsă legală aspră, inclusiv cu închisoarea pentru conducerea companiei, în cazul firmelor găsite vinovate de încălcarea confidențialității on-line.

Pe de altă parte, dacă un utilizator accesează Web-ul de pe mai multe calculatoare, o bază de date server-side poate să simplifice viața utilizatorului, deoarece acesta nu va trebui să introducă datele din nou pe fiecare calculator. Totuși, este necesară o secvență iritantă de login și furnizare de parolă înainte de a putea accesa situl. Vom discuta această problemă mai amănunțit în secțiunea următoare.

ID-uri și parole utilizator

O tehnologie actuală pe care aproape toată lumea o urăște este ID-ul și parola de utilizator, care are avantajul că pune problema identificării utilizatorului direct în mâinile acestuia. Dacă un utilizator nu mai dorește să fie identificat prin acel ID, poate pur și simplu să nu îl mai folosească. Și, desigur, parolele și ID-urile de utilizator sunt cruciale pentru anumite tipuri de sisteme sigure. De fapt, unele niveluri de securitate nu pot fi realizate în mod rezonabil fără a cere perechi ID/parolă de utilizator.

Dar majoritatea schemelor existente ignoră cerințele de securitate reală a parolelor. Deși un sit poate cere o parolă, de obicei nu oferă nici un mijloc prin care aceasta să fie modificată dacă se află că a fost compromisă, iar restricțiile impuse asupra formatului parolelor sunt minime sau chiar inexistente. Deci cuvintele de dicționar sunt acceptate în majoritatea sistemelor, ca de altfel și folosirea aceleiași parole ani la rând, sau a unor parole ridicol de scurte sau evidente.

Slăbiciunea fatală a fiecărui sistem de parole specific sitului este aceea că se cer un ID și o parolă separate pentru fiecare sistem nou, deci utilizatorul ajunge la un număr de parole

și ID-uri mai mare decât poate să țină minte, iar situl trebuie să recicleze ID-urile de utilizator în funcție de timpul care a trecut de la crearea lor. Deci, un utilizator care are o parolă corectă se trezește pe dinafară, situl cedând ID-ul său altcuiva, deoarece utilizatorul nu a vizitat situl destul de des pentru a satisface organizația.

Este o sursă nesfârșită de frustrare pentru utilizatori. Fie își vor lipi pe pereți etichete autocolante cu ID-uri și parole, fie vor încerca să folosească același ID și aceeași parolă pentru fiecare sit, o breșă semnificativă de securitate. Ne putem imagina că un administrator fără scrupule al unui sit care colectează ID-uri și parole de utilizator poate folosi ID-ul și parola de utilizator pe alte situri, având destul de mari șanse de succes. Acest lucru este rău sau extrem de rău, în funcție de siturile pe care le-a vizitat utilizatorul.

Sugestie de la LeeAnne.com Words to weave by

Multe situri cer acum explicit utilizatorilor să folosească drept ID adresa lor de e-mail. Din punctul de vedere al sitului, acesta este un avantaj deoarece obține adresa de e-mail. Utilizatorul poate să țină minte ușor ID-ul și poate folosi peste tot același ID și aceeași parolă. Din punctul de vedere al securității, este o strategie deplorabilă deoarece jumătate din perechile ID/parolă de utilizator sunt ghicite ușor de hackeri, înlăturând astfel o parte din protecția dumneavoastră. Iar probabilitatea de a utiliza același ID și aceeași parolă peste tot presupune că angajații oricărui sit pot folosi informația dumneavoastră pentru a se conecta în numele dumneavoastră oriunde.

Deci, parolele, deși teoretic pot asigura niveluri extraordinare de securitate dacă sunt folosite corect, au în practică slăbiciuni atât de mari încât siguranța lor este în general îndoielnică. Doar utilizatorul cel mai meticulos și mai hotărât poate obține întreaga securitate posibilă cu o schemă de parolă, iar utilizatorii ocazionali tind să folosească parole de o siguranță minimă, dar ușor de memorat.

P3P – Platforma pentru preferințe de confidențialitate

P3P este la momentul actual în ultimul stadiu de schiță de lucru și este programată pentru trecerea la stadiul final de recomandare în iunie 2000. Deși acest lucru se poate modifica, probabil în direcția unor întârzieri, standardul existent este destul de aproape de ceea ce este necesar și va evolua puțin sau deloc.

Un lucru care în mod sigur se va modifica este mecanismul de transfer de date, care, conform grupului de lucru, va fi șters din specificație la următoarea revizuire. S-a determinat că majoritatea producătorilor aveau de gând să folosească standarde pentru transfer de date, brevetate sau deschise, deci nu prea era nevoie de un mecanism separat.

De fapt, singura justificare pentru păstrarea sa ar fi comoditatea utilizatorului care, altfel, va fi nevoit să introducă datele din nou pentru fiecare schemă brevetată. Omniprezența cozilor lungi la telefoane și dificultatea de utilizare a sistemelor automate de telefonie pot fi un indicator de încredere al ideilor fixe și al lipsei de preocupare reală pentru comoditatea utilizatorilor ale majorității corporațiilor și ale organizațiilor Internet.

Oricât am insista, dificultatea specificării unui mecanism de transfer de date fără a neglija sau a omite aspecte importante de securitate nu poate fi exagerată. Experții în securitate și telecomunicații au petrecut ani proiectând și verificând astfel de standarde. A sugera

că un standard nou poate fi definit în câteva luni este o aberație. Mult mai bine este să cerem părerea specialiștilor în criptare și securitate care au mulți ani de experiență și să recurgem la produse existente, decât să facem o treabă de mântuială, compromițând standardul P3P ca întreg sau întârziind apariția acestui standard până la momentul în care un astfel de mecanism ar putea fi proiectat în mod robust.

Ce se întâmplă cu P3P

Standardul P3P este un protocol și, indirect, un standard de colectare a datelor. Prima interacțiune pe care o are utilizatorul cu un browser cu suport P3P este un tip de cerere de date. Pentru fiecare fragment de date, browserul ar trebui să afle modul în care utilizatorul va permite să fie folosită informația respectivă. P3P nu specifică modul în care se colectează și se stochează aceste date, deci browserul este liber să implementeze o casetă de dialog interactivă, un ecran cu opțiuni la intervenția utilizatorului sau chiar să colecteze fiecare informație ca răspuns la propunerile browserului. Aceasta este secvența evenimentelor care constituie o negociere P3P.

Mai întâi, clientul efectuează o cerere GET:

```
GET /index.html HTTP/1.1
Host: www.leeanne.com
Accept: */*
Accept-Language: en
User-Agent: NewBrowser/6.0
```

În acest moment, clientul nu știe dacă situl are sau nu stabilită o politică de confidențialitate. Acest lucru se clarifică numai atunci când serverul returnează un răspuns care indică un standard de confidențialitate și o descriere în XML a politicii de confidențialitate a sitului care poate fi citită de calculator:

```
HTTP/1.1 200 OK
Opt: "http://www.w3.org/2000/P3Pv1"; ns=12
12-Policy: http://www.leeanne.com/usingxml/14xmp01.xml
Content-Type: text/html
Content-Length: 1500
Server: NewServwe/2.0
```

Clientul poate apoi să compare politica returnată cu preferințele utilizatorului, să decidă dacă politica sitului este acceptabilă și să întrerupă transferul în caz contrar. Agentul poate apoi să întrebe utilizatorul dacă acceptă grade mai mici de confidențialitate, afișând toate datele CONSEQUENCES pentru a ușura decizia utilizatorului și afișând, dacă este posibil, politica sitului într-o formă lizibilă pentru a clarifica subiectele implicate. Dacă utilizatorul decide că o confidențialitate mai slabă este acceptabilă, atunci transferul continuă. În caz contrar, transferul este încheiat.

Dacă situl nu are nici o politică, clientul presupune că este fixată cea mai slabă politică de confidențialitate posibilă și întreabă utilizatorul dacă dorește să continue. Încurajăm utilizatorii să părăsească imediat situl, deoarece singura modalitate prin care putem constrânge operatorii siturilor Web să implementeze politicile P3P este să micșorăm numărul de accesări ale siturilor lor.

Organizațiile care utilizează P3P vă vor trimite o propunere similară la prima dumneavoastră vizită, descriind informațiile pe care le doresc de la dumneavoastră, precum și ce au de gând să facă cu ele. Propunerea va explica ce se întâmplă dacă le acordați acces

la date și ce se întâmplă în caz contrar. Organizația este liberă să își fixeze orice politică dorește. Utilizatorul este liber să ia orice decizie, dar ar trebui să păstreze toate URI-urile de politică astfel încât să le poată compara cu URI-ul returnat la vizita următoare pentru a afla dacă politica a fost sau nu schimbată.

Dacă propunerea este acceptată, utilizatorul probabil nici nu va sesiza că tranzacția a avut loc. Dacă a fost respinsă, browserul poate să afișeze o casetă de dialog care să ceară o decizie pentru a permite un acces mai larg la datele confidențiale ale utilizatorului pentru situl respectiv. O slăbiciune a limbajului P3P în forma sa actuală este că presupune că singurul participant la interacțiune care poate fi flexibil în ceea ce privește confidențialitatea este utilizatorul. Situl are o flexibilitate minimă în negocierea parametrilor de confidențialitate pentru vizită, având numai aptitudinea rudimentară de a stabili politici diferite pentru subdirectoare diferite.

Deoarece mecanismul intern de transfer de date a fost înlăturat din specificația P3P, este clar că transferul de date va fi realizat printr-un proces extern. Este foarte logic. Complexitatea deplină a unui protocol care suportă internaționalizare, accesibilitate, date simple și structurate, semnături digitale, actualizări periodice, termene de valabilitate și o mulțime de alte probleme devine evidentă atunci când chestiunea este investigată în detaliu.

Formatul de transfer al politicilor de confidențialitate

P3P este încă în stadiul de schiță de lucru, iar la momentul actual nu există aplicații care să suporte complet standardul. Ultima variantă a schiței este planificată pentru sfârșitul lui aprilie 2000 și ar trebui să treacă la stadiul de candidat de recomandare imediat după această dată. Până atunci ar fi prematur să vă bazați pe orice în afara tendințelor generale ale standardului, deoarece a mai ajuns odată în faza finală și a fost trimis înapoi pentru revizuire.

Standardul final va semăna probabil cu profilul P3P listat în secțiunea „Studii de caz" de la sfârșitul acestui capitol, un element POLICY care include atât elemente obligatorii, cât și elemente opționale.

Elementul POLICY în sine trebuie să conțină un atribut xmlns care să indice versiunea P3P utilizată și un atribut ENTITY care să conțină numele entității care descrie politica de confidențialitate a sitului și informații de contact care să descrie modul în care îl puteți contacta.

Elementul POLICY ar trebui să conțină un element DISCLOSURE care să descrie locația politicii de confidențialitate, într-o formă lizibilă, asociată cu politica în format XML și o descriere sumară a celor care au acces la informația colectată.

Elementul POLICY ar trebui de asemenea să conțină un element DISPUTES-GROUP format din elemente DISPUTES care descriu ce se face în cazul unor dispute referitoare la confidențialitatea datelor dumneavoastră. Aceste elemente pot indica orice, de la servicii pentru clienți ale sitului, la organizații de asigurare a confidențialității și până la tribunale pentru procese civile.

Apoi, elementul POLICY ar trebui să conțină unul sau mai multe elemente STATEMENT care să descrie tipurile efective ale datelor colectate și utilizările pe care le vor primi. Elementul STATEMENT conține elemente PURPOSE, un element RECIPIENT, un element DATA-GROUP (care la rândul său conține elemente DATA), un element RETENTION și un element opțional CONSEQUENCE-GROUP.

Elementul PURPOSE descrie scopul în care va fi utilizată informația. Elementul RECIPIENT îi spune utilizatorului cine poate folosi informația. Elementul DATA-GROUP conține unul sau mai multe elemente DATA care descriu datele colectate. Elementul RETENTION îi spune utilizatorului cât timp vor fi păstrate datele. Elementul CONSEQUENCE-GROUP conține unul sau mai multe elemente CONSEQUENCE, care îi spun utilizatorului care sunt beneficiile ce decurg din permiterea colectării acestor date. Deoarece aceste informații pot fi afișate în mod prestabilit pentru utilizatori sau ca răspuns la o casetă de dialog referitoare la confidențialitate, elemente multiple CONSEQUENCE pot primi atribute diferite `xml:lang` pentru a permite localizarea consecințelor.

Dacă studiați exemplul dat în secțiunea „Studii de caz" care conține o politică de confidențialitate în text simplu și în echivalentul său P3P, atunci totul va deveni foarte clar. În momentul în care standardul actual va fi publicat ca recomandare, îl veți găsi la `http://www.w3.org/p3p/`.

Rezultate finale ale confidențialității

Politicile P3P promit un control destul de bun al confidențialității dumneavoastră, dacă va putea fi rezolvată tensiunea dintre operatorii de situri Web și utilizatori. Dar W3C a realizat că unii operatori pot să promită o confidențialitate deplină, dar să nu își ducă promisiunea până la capăt sau să o încalce. În fond, au o grămadă de date care valorează potențial milioane de dolari. Chiar dacă au intenții bune, capriciile ciclurilor de afaceri pot să îi tenteze să își schimbe ulterior politicile sau chiar să vă mintă în față.

Deci elementele de siguranță identifică un anumit tip de organism de auditare care examinează și verifică dacă politicile sunt conforme cu afirmațiile proprietarului și se asigură că nu pot fi modificate de la o vizită la alta fără un preaviz.

Sugestie de la LeeAnne.com Words to weave by

Este aproape imposibil pentru utilizatori să afle despre o încălcare a confidențialității lor dacă nu folosesc obsesiv ID-uri diferite pentru fiecare sit și dacă nu păstrează cu îndârjire înregistrări. Majoritatea se vor baza pe brokeri de servicii de confidențialitate, care primesc un stimulent financiar pentru a păstra un anumit nivel de control asupra informației, și pe firme de examinare cum este TRUSTe, de la `http://www.truste.org/`.

Stabilirea faptului că o politică nu s-a modificat este o parte vitală, chiar dacă minimă, a standardului P3P. În caz contrar ar fi posibil ca utilizatorii să fie momiți cu o reclamă de politică de confidențialitate strictă, să se colecteze numele și ceilalți identificatori, să se modifice politica în „publică" și să se vândă informațiile. Deci P3P susține că URI-ul politicii în sine trebuie să se modifice dacă politica se modifică. În acest mod, măcar agentul dumneavoastră utilizator vă poate spune că politica s-a modificat de când ați vizitat ultima dată situl.

Dar vor exista și jucători mai puțin importanți, care vor forța limitele moralității dacă nu îi supraveghează nimeni. Deci agențiile de examinare externe vor fi la un nivel destul de proeminent în lumea nouă a confidențialității. Printre furnizorii actuali de garanții sunt BBB Online și TRUSTe, acestea furnizând sigilii care pot fi plasate pe un sit pentru a asigura utilizatorii într-un mod ostentativ că acel sit are grijă de confidențialitatea lor. Dar am văzut deja că până și aceste protecții minimale sunt destul de ușor de ocolit în climatul politic și economic actual din SUA. Situația confidențialității în Europa este mult mai bună, iar Uniunea Europeană este în avangarda legislației referitoare la confi-

dențialitate pe plan mondial, ceea ce în cele din urmă s-ar putea să confere avantaje economice firmelor europene dacă utilizatorii vor deveni într-adevăr preocupați de confidențialitatea lor on-line.

Deci, brokerii de servicii de confidențialitate și agențiile de examinare pot să prevină practicile abuzive sau cel puțin vă dau posibilitatea să le evitați, de exemplu siturile care se maschează în ceva complet diferit pentru motoarele de căutare, iar apoi deschid spontan zeci de ferestre spre alte situri pentru a-și mări veniturile „clickthrough" (venituri în funcție de numărul de clicuri). Totuși, nu există nici o schemă actuală care să aibă legea în spatele ei, deci majoritatea sunt inofensive. O dată cu răspândirea tot mai largă a mecanismelor de plăți minime apare potențialul de a escroca utilizatorii, făcând browserele lor să sară printr-o serie fără sfârșit de pagini minuscule pay-per-view. Dacă valoarea fiecărei tranzacții în parte este destul de mică, mulți utilizatori își vor seta portofele electronice care să plătească automat un cent, să zicem, în loc să decidă individual dacă o pagină sau alta merită o sutime de dolar.

Pe de altă parte, în ciuda unor practici oarecum abuzive din partea unor situri, furnizorii de garanții au avut rețineri în a-și retrage sigiliile de pe siturile care au cam forțat limitele. Acest lucru scoate la iveală tensiunea dintre angajamentele agențiilor de garantare și dorința lor perfect naturală de a continua să fie plătite.

OPS/P3P – Standardul deschis pentru profiluri (Open Profiling Standard)

În 1997, Netscape Communications Corporation, Firefly Network, Inc. și Verisign, Inc. au propus către W3C *Open Profiling Standard* (OPS). Ei și-au imaginat că standardul va conlucra cu P3P, pe care îl va completa, și s-ar putea să se întâmple chiar așa. Microsoft Corporation a înaintat de asemenea un standard, *Privacy and Profiling on the Web* (*confidențialitate și profiluri pe Web*).

W3C a luat în considerare aceste propuneri atunci când a dezvoltat P3P și a încorporat unele elemente – mai mult conceptele simple: informațiile despre utilizator și negocierile cu serverul – în protocolul P3P. Nu a fost inclus totul, în particular criptarea și mecanismele de semnătură. Semnăturile digitale W3C (W3C Digital Signatures) pot să compenseze această lipsă completând câmpul `certificate`, dar decizia grupului de lucru P3P de a elimina mecanismul de transfer de date va împinge mecanismele de criptare spre furnizori terți, împreună cu transferul de date. De asemenea, au eliminat personajele multiple din Privacy and Profiling on the Web, care erau o îmbunătățire semnificativă a confidențialității pentru mulți utilizatori deoarece le permiteau să creeze multiple identități false, putând astfel să navigheze pe situri fără a fi puși în corespondență cu celelalte personaje.

P3P se va modifica fără îndoială până în momentul în care veți citi aceste rânduri și va încorpora cârlige spre instrumente de transfer de date de la terțe părți. OPS sau un descendent al acestuia poate fi unul dintre ele.

JavaScript și limbaje de scriptare

Nu există nici un motiv teoretic pentru care P3P să nu poată fi implementat cu o casetă de dialog Java, sau chiar JavaScript, înglobată într-o pagină descărcată de pe un sit Web. Din perspectiva proprietarului sitului se economisesc resurse pe server, deoarece între-

gul dialog al protocolului, inclusiv negocierea și colectarea datelor, poate fi realizat fără a fi necesară o prelucrare pe server. Singura problemă este că trebuie să existe o modalitate standard de a stoca datele pe calculatorul utilizatorului, în caz contrar acesta trebuind să introducă iar și iar aceeași informație.

În acest moment nu există un astfel de mecanism. Cookie-urile sunt specifice pentru un anumit sit, iar permisiunea ca procese externe să citească și să scrie pe calculatoarele utilizatorilor este o nesăbuință din partea acestora. Microsoft încarcă cu regularitate alerte și actualizări critice de securitate pe situl lor de actualizare, în principal din cauza slăbiciunilor din interfața OLE/ActiveX care permit hackerilor accesul de citire și scriere pe calculatoarele utilizatorilor prin oricare din miile de breșe care le-au scăpat programatorilor.

În combinație cu un anumit tip de proces utilizator, cum ar fi programul Java PrivacyMinder, descris în cele ce urmează, abordarea client-side a negocierii politicii pare să aibă sens, dar în orice caz această schemă va fi în curând eliminată prin suportul nativ în browser, deci este greu de văzut care sunt beneficiile sale reale.

Există deja implementări P3P, dar numai ca prototip. PrivacyMinder de la AT&T este un program Java client-side care permite transferul de date personale stabilite de utilizator. Nivelul de securitate prestabilit din PrivacyMinder este într-adevăr foarte privat, deoarece nu permite colectarea nici unei informații identificabile, monitorizează cererile de informații de la depozitul de date al utilizatorului și blochează cererile care încalcă setările utilizatorului.

Netscape (Mozilla) va implementa mai mult ca sigur P3P, deoarece P3P este pe lista lucrurilor dorite de Netscape și poate fi implementat în cea mai mare parte ca aplicație a RDF, care în mod sigur va fi disponibil. În momentul de față, P3P nu este deloc prezent în codul dezvoltării. Este logic, deoarece standardul este încă o schiță de lucru

Microsoft are de gând să ofere suport pentru P3P, probabil imediat după finalizarea standardului. Portofelul electronic Microsoft Passport are deja interfețe cu TRUSTe, `http://www.truste.org/`, și cu BBB Online, `http://www.bbbonline.com/`, situl Better Business Bureau.

Browserul Opera are o abordare inovatoare pentru completarea formularelor, jumătate din problema plictisitoare a furnizării sau a blocării accesului la datele utilizatorului. Opera păstrează o bază de date internă cu informații despre utilizator, care poate fi utilizată pentru a completa formulare printr-o apăsare de tastă sau două pentru fiecare câmp. Această caracteristică poate fi utilizată pentru a ușura completarea manuală a formularelor, o modalitate de a controla unde ajung informațiile. Iar Opera are și posibilitatea de a bloca rularea limbajelor de scriptare și poate să șteargă automat cookie-uri de fiecare dată când închideți browserul, deci un anumit nivel de confidențialitate este asigurat chiar la nivelul browserului.

Sugestie de la LeeAnne.com Words to weave by

Dacă sunteți destul de norocos să deținеți propriul domeniu și aveți autoritate de postmaster, deci primiți întreaga corespondență, un mod de a verifica propria dumneavoastră confidențialitate este să completați adresa de e-mail cu informații despre numele de utilizatori, identificând situl la care le oferiți. Dacă acest ID apare mai târziu într-un spam de la alt sit, veți ști pe cine să acuzați atunci când primiți e-mail-uri trimise la *neserios.sitweb*`@domeniulmeu.com`.

Studii de caz

Să presupunem că doriți să creați o declarație de confidențialitate pentru situl meu Web, `http://www.leeanne.com/`.

Nu doriți să angajați un avocat, deoarece situl nu colectează de fapt nici o informație și nici nu face nimic cu nici o informație, nici măcar nu studiază datele de jurnal de pe server care ar fi disponibile dacă am vrea să ne uităm la ele. Microsoft LinkExchange dispune de un asistent on-line pentru confidențialitate, care vă ajuta cu un model de declarație de confidențialitate lizibilă, la `http://privacy.linkexchange.com/`. Veți răspunde la o serie de întrebări referitoare la scopul în care veți folosi informațiile de pe sit, inclusiv politicile siturilor la care vă legați și alte câteva chestiuni mărunte. Putem vedea textul rezultat în Figura 15.1.

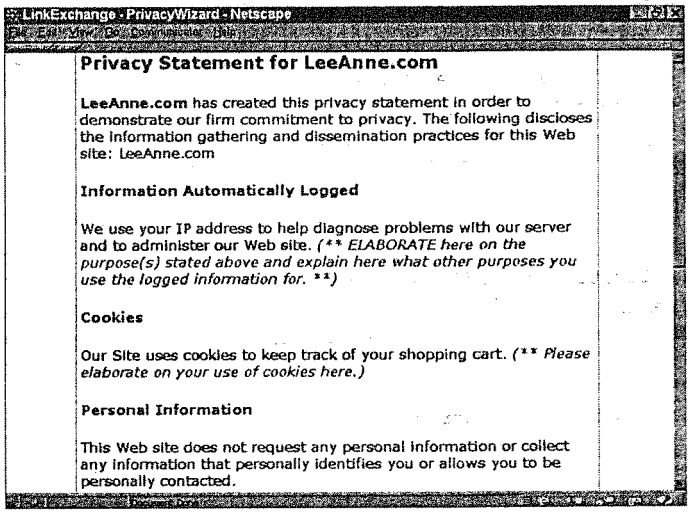


FIGURA 15.1
Microsoft LinkExchange Privacy Wizard a generat această declarație de confidențialitate lizibilă.

Următoarele informații de confidențialitate de la asistent sunt listate într-un format de text simplu, astfel încât să le puteți vizualiza cu ușurință. Locurile care trebuie completate de utilizator au fost lăsate așa cum apar, ca să știți unde le plasează asistentul:

```
Privacy Statement for LeeAnne.com
LeeAnne.com has created this privacy statement in order to demonstrate our firm
commitment to privacy. The following discloses the information gathering and
dissemination practices for this Web site: LeeAnne.com
Information Automatically Logged
We use your IP address to help diagnose problems with our server and to administer
our Web site. (** ELABORATE here on the purpose(s) stated above and explain here
what other purposes you use the logged information for. **)
Cookies
Our Site uses cookies to keep track of your shopping cart. (** Please elaborate on
your use of cookies here.)
Personal Information
This Web site does not request any personal information or collect any information
```

that personally identifies you or allows you to be personally contacted.
Because we do not collect any personal information on this Web site, we do not share any personal with any third parties nor do we use any personal information for any purposes.

External Links

This site contains links to other sites. LeeAnne.com is not responsible for the privacy practices or the content of such Web sites.

Public Forums

This site makes chat rooms, forums, message boards, and/or newsgroups available to its users. Please remember that any information disclosed in these areas becomes public information and you should exercise caution when deciding to disclose your personal information.

Security

This site has security measures in place to protect the loss, misuse, and alteration of the information under our control. (** Please elaborate on the security measures that you have in place. **)

Contacting the Web Site

If you have any questions about this privacy statement, the practices of this site, or your dealings with this Web site, you can contact:
info@leeanne.com

adică

Declarație de confidențialitate pentru LeeAnne.com

LeeAnne.com a creat această declarație de confidențialitate pentru a demonstra angajamentul nostru ferm în privința confidențialității. În cele ce urmează sunt prezentate practicile de colectare și distribuire a informațiilor pentru acest sit Web: LeeAnne.com

Informații înregistrate automat

Utilizăm adresa dumneavoastră IP pentru a diagnostica probleme legate de serverul nostru și pentru a administra situl nostru Web. (** DETALIAȚI aici scopul (scopurile) prezentate mai sus și explicați aici alte scopuri în care este utilizată informația înregistrată. **)

Cookie-uri

Situl nostru utilizează cookie-uri pentru a urmări lista dumneavoastră de cumpărături. (** Vă rog detaliați aici modul în care utilizați cookie-urile.)

Informații personale

Acest sit Web nu cere nici o informație personală și nu colectează nici o informație care vă identifică personal sau permite să fiți contactat personal.
Deoarece nu colectăm nici o informație personală pe acest sit Web, nu partajăm informații personale cu terțe părți și nici nu utilizăm în nici un scop informații personale.

Legături externe

Acest sit conține legături la alte situri. LeeAnne.com nu răspunde pentru practicile de confidențialitate sau pentru conținutul acestor situri Web.

Forumuri publice

Acest sit organizează camere de conversație, forumuri, aviziere și/sau grupuri de informare pentru utilizatorii săi. Vă rog să rețineți că orice informație dezvăluită în aceste locuri devine informație publică și ar trebui să fiți atent atunci când vă decideți să divulgați informații personale.

Securitate

Acest sit are implementate măsuri de securitate pentru a proteja informațiile aflate sub controlul nostru împotriva pierderii, folosirii incorecte și modificării. (** Vă rog detaliați aici măsurile de securitate implementate. **)

Contactarea sitului Web

Pentru orice întrebări referitoare la această declarație de confidențialitate, la practicile acestui sit sau la utilizarea acestui sit Web, puteți contacta:
info@leeanne.com

Deoarece nici un asistent nu poate să prevadă toate posibilitățile, acesta va genera mai multe spații de completat pentru a arăta unde trebuie extinsă declarația. Desigur, dacă doriți să menționați mai multe utilizări ale datelor pe care le colectați, în principal adrese IP și alte informații tehnice, le veți adăuga aici. Puteți presupune că o adresă IP nu poate duce la identificarea unei persoane, deoarece majoritatea celor care navighează prin intermediul unui ISP partajează o rezervă comună de adrese. Totuși, mulți utilizează DSL și alte tehnologii avansate de conectare care adesea au o adresă de IP individuală atribuită unui utilizator individual. În aceste cazuri, o adresă IP este o linie directă la calculatorul utilizatorului și trebuie păzită cu strășnicie.
Deci înainte de a promite că situl nu colectează date identificabile și nu o va face nici în viitor, ar trebui să vă gândiți serios la toate implicațiile acestei afirmații. În cazul acestui sit nu este planificată nici o interactivitate, dar nu se știe niciodată. În viitor, poate că se va dori să se adauge facilitatea de a cumpăra cărți on-line sau de a permite afișarea de comentarii pe un avizier on-line.
Deși această politică de confidențialitate generată este un punct de pornire, ar trebui să testați câteva iterații cu presupuneri diferite, înainte de a stabili o politică restrictivă pentru un sit. Nu uitați că politicile P3P ale unui anumit sit ar trebui să rămână stabile de-a lungul timpului dacă dorim ca această schemă care se autoreglează să aibă vreun înțeles. Chiar dacă nu credeți că vă veți răzgândi vreodată referitor la politicile sitului dumneavoastră, moștenitorii și împuterniciții dumneavoastră pot să aibă idei diferite. Dacă situl dumneavoastră face parte dintr-o întreprindere mai mare sau dintr-o strategie de marketing, atunci ar trebui neapărat să contactați un avocat căruia să-i cereți sfaturi înainte de a afișa un astfel de document pe Web.
Iată un exemplu de fișier XML P3P generat pentru a se conforma mai mult sau mai puțin textului anterior al politicii de confidențialitate:

```
<POLICY xmlns="http://www.w3.org/2000/P3Pv1"
        entity="LeeAnne.com, 4200 Park Boulevard #250,
                Oakland, CA, USA"> .
  <DISPUTES-GROUP>
    <!-- The following three lines could be inserted to
         point to an assurance organization that would
         audit the privacy policies of this site. Since
         this is only an example, I'll point to an
         organization guaranteed to do nothing. -->
    <DISPUTES resolution-type="independent"
              description="LeeAnne.org"
              service="http://www.leeanne.org"
```

```
      image="http://www.leeanne.com/usingxml/14xmp01.gif"/>
   </DISPUTES-GROUP>
   <DISCLOSURE
      discuri="http://www.leeanne.com/usingxml/14xmp01.html"
      access="nonindent" change_agreement="no" />
   <STATEMENT>
      <IDENTIFIABLE><no /></IDENTIFIABLE>
      <CONSEQUENCE-GROUP>
        <CONSEQUENCE>You're able to use our shopping cart
                     to order merchandise</CONSEQUENCE>
        <RECIPIENT><ours /></RECIPIENT>
        <PURPOSE><current /></PURPOSE>
        <!-- The following information is here because we
             may, at some future date, implement a shopping
             cart service for online ordering. At the present
             time we neither set nor read cookies or gather
             any other data. -->
        <DATA-GROUP>
          <DATA name="dynamic.cookies" category="state" />
          <DATA name="dynamic.miscdata" category="state" />
        </DATA-GROUP>
        <RETENTION><indefinitely /></RETENTION>
      <STATEMENT>
      <IDENTIFIABLE><no /></IDENTIFIABLE>
        <RECIPIENT><ours /></RECIPIENT>
        <PURPOSE><admin /><develop /></PURPOSE>
        <!-- The following information is collected
             automatically by the server. Although we
             don't do anything with it at the present
             time we may, at a later date, use this
             data to optimize our server, Web site, or
             ISP connectivity options. -->
        <DATA-GROUP>
          <DATA name="dynamic.clickstream.server" />
          <DATA name="dynamic.http.useragent" />
          <DATA name="dynamic.http.referrer" />
        </DATA-GROUP>
        <RETENTION><indefinitely /></RETENTION>
      <STATEMENT>
</POLICY>
```

Corespondența între textul lizibil și codul XML este în mod inevitabil aproximativă, deoarece limbajul natural este un mediu imperfect, chiar și în cel mai bun caz. În cazul unei declarații de confidențialitate efective, ar fi înțelept să consultați atât un avocat cât și un programator, pentru a afla cum se potrivesc aceste două declarații.

Securitate și confidențialitate

CAPITOLUL 16

În acest capitol

- Date vulnerabile și servere de date pe Web
- Modul în care ruterele protejează datele dumneavoastră
- Construirea unui firewall de securitate
- Crearea de parole unice
- Criptare
- Inițiative de securitate și confidențialitate
- Depanarea problemelor de securitate

Date vulnerabile și servere de date pe Web

XML nu este un sistem de protocoale sigur. De fapt, a fost proiectat să fie perfect deschis și transparent pentru examinare. Deci dacă aveți nevoie de securitate, iar de aceasta are nevoie aproape oricine, trebuie să adăugați caracteristici de securitate din afara lumii XML. Chiar și inițiativele de securitate legate de XML, cum sunt semnăturile XML (XML Signatures) și P3P, sunt text simplu de meta-descriere a securității, și nu implementări efective.

Totul din ciberspațiu devine, cu fiecare minut, tot mai conectat. Din ce în ce mai multe date financiare și medicale particulare și alte date personale sunt stocate pe Web în formate accesibile. Evident, cele mai accesibile date personale sunt informațiile de pe propriile noastre calculatoare, deoarece majoritatea calculatoarelor de acasă și din companiile mici au o securitate care nu merită menționată. După acestea, urmează informațiile stocate pe serverele organizațiilor importante din domeniul financiar și al serviciilor medicale.

Informațiile de pe serverele serviciilor financiare și medicale sunt de obicei bine protejate împotriva hackerilor cei mai hotărâți, deoarece o bancă vulnerabilă își poate vedea buzunarele electronice jumulite de milioane de dolari cât ai clipi din ochi. O organizație de servicii medicale vulnerabilă poate ajunge în aceeași situație ca urmare a unui proces prelungit, intentat pentru divulgarea întâmplătoare a informațiilor private.

Totuși, majoritatea acestor firme pot vinde în mod legal informațiile pe care le dețin, unor terțe persoane, fără a vă anunța. Companiile de asigurări fac schimburi de date între ele în mod firesc, spun ei, „pentru a combate frauda". Dar o politică similară, de spionare a milioane de oameni nevinovați în speranța prinderii unui escroc de ocazie ar fi evident neconstituțională dacă ar fi făcută de poliție. Instituțiile financiare fac același lucru, afișează istoria tranzacțiilor tuturor clienților pe aviziere virtuale unde oricine are destui bani de cheltuit poate să o vadă. Până și farmaciile au fost prinse vânzând numele

persoanelor care utilizează anumite medicamente companiilor producătoare de medicamente, care pot apoi să vândă produsele concurente direct persoanelor interesate.

În ciuda faptului că pentru unii acest lucru poate părea destul de nevinovat, există suficiente medicamente sau combinații de medicamente care au scopuri bine definite și pot dezvălui mult mai multe decât ar dori oamenii să știe *oricine* despre ei și, cu atât mai puțin, un comerciant veros care are interese economice obscure. Sau, mai rău, un oponent politic, un șef sau oricine altcineva care are puterea să ne facă rău dacă află câteva din secretele noastre. Deci o persoană care are o rețetă pentru medicamente antipsihotice, pe care le folosește pentru a-și trata schizofrenia, poate fi dezavantajată în cazul unei angajări dacă s-ar cunoaște acest lucru. Altcineva ar putea fi preocupat de faptul că orice străin poate afla că ia antidepresive sau antibiotice prescrise deseori pentru herpesul genital și, de asemenea, pentru infecțiile herpes sau cu alți viruși care apar la pacienții bolnavi de SIDA.

Siturile de informații despre sănătate, care obțin venituri din publicitate, pot genera date foarte personale și pentru furnizorul de spațiu de reclamă, deoarece acesta poate să afle legăturile selectate de utilizator și să comunice advertiserilor tipul de pagini care îl interesează pe utilizator. Selectați informații despre căderea părului și vor apărea reclame la Rogaine[1]. Poate veți începe să primiți oferte prin poștă, sau chiar prin telefon.

În trecut, unele date aveau atașat un nivel minim de securitate deoarece formatele erau foarte comprimate și, în cazul bazelor de date relaționale, foarte greu de descifrat dintr-o grămadă de înregistrări brute. Deci, înregistrarea unui pacient dintr-un spital, într-o bază de date moștenită, poate arăta astfel:

```
Jane Doe/789-1234567/047/398 211 734/ ...
```

Toate informațiile partajate de mai mulți pacienți sunt indexate în tabele secundare, deci descifrarea unei înregistrări necesită un nivel de experiență în baze de date care poate depăși aptitudinile marii majorități a utilizatorilor bazei de date.

O versiune XML a informației trecute criptic în exemplul anterior este mult mai accesibilă privirilor indiscrete:

```
<pacient>
  <nume>Jane Doe</nume>
  <ID-medical>789-1234567</ID-medical>
  <alergii>penicilina</alergii>
  <medicatie>
    o combinatie de medicamente extrem de stanjenitoare
  </medicatie>
  <istoric>
    <inregistrare>
      informatii si mai stanjenitoare
    </inregistrare>
    <inregistrare>
      secrete pe care nu le stie nici mama dumneavoastra
    </inregistrare>
  </istoric>
</pacient>
```

În acest formular puteți vedea mult mai clar de ce poate fi cineva îngrijorat. Având destule informații despre medicamentele luate de cineva, oricine poate trage concluzii evidente despre viața particulară a altcuiva chiar fără a studia notele clinice sau alte

[1] N.T. – Un medicament pentru prevenirea căderii părului.

înregistrări. Deoarece înregistrările sunt inteligibile pentru oricine, trebuie să cercetați toate modurile posibile de accesare a înregistrării și să le blocați.

Disponibilitatea, care va exista curând, de fluxuri nesfârșite de date prin Internet utilizând fluxuri de transmisie bazate pe XML, va face controlul informațiilor personale chiar mai important, deoarece așa cum am văzut în înregistrarea anterioară, semnificația fluxului este evidentă dacă nu se iau măsuri de cifrare a acestuia în timpul transmisiei.

În Capitolul 15, „Obținerea informațiilor despre utilizator", ați studiat P3P, platforma de confidențialitate a W3C, care încearcă să ofere utilizatorilor măcar un pic de control asupra datelor lor personale. Totuși, P3P nici măcar nu încearcă să acopere toate problemele de securitate, deoarece vă protejează numai împotriva celor care joacă respectând regulile. Există grupuri de confidențialitate electronică, cum este EPIC, la `http://www.epic.org/`, care încearcă să impună un control mai mare asupra confidențialității. Federal Trade Commission (Comitetul Federal de Comerț al SUA) și Uniunea Europeană încearcă să promoveze regulamente, în parte pentru a ajuta la crearea unei legături între legile de confidențialitate foarte restrictive din Europa și situația confidențialității on-line din SUA, care este relativ lipsită de un control legal. Dar acestea afectează deocamdată numai firmele și persoanele care acordă atenție legilor și regulamentelor.

În acest capitol veți studia modul în care să vă ocupați de cei care nu respectă nici o regulă, învățând cum poate fi protejată în timpul stocării, transmisiei sau actualizării această înregistrare a unui pacient sau alte date sensibile asemănătoare.

Sugestie de la LeeAnne.com Words to weave by

Pentru majoritatea siturilor, principalul mijloc de prevenire a intruziunilor ar fi un lacăt pe ușa încăperii cu calculatoare, deoarece multe furturi sau interceptări de date nu necesită decât acces la sistemul fizic și poate una sau două dischete.

Controlul fizic al accesului la înregistrare este prima metodă de apărare deoarece înregistrarea fizică este, într-o oarecare măsură, principală și poate fi mult mai detaliată decât o înregistrare de pe calculator legată la dosarul fizic. Radiografiile, de exemplu, sunt rareori digitizate în cabinete, deci o înregistrare medicală on-line indică numai la locația fizică a dosarului care le conține.

De exemplu, un sistem de înregistrări on-line care este conectat la Internet pentru ca utilizatorii săi să poată trimite e-mail-uri este de asemenea vulnerabil la intruziunile electronice din exterior dacă nu se iau măsuri de prevenire.

Modul în care ruterele protejează datele dumneavoastră

Dacă o rețea internă este conectată la Internet, atunci oricine din exterior poate accesa sistemele din rețeaua locală, în cazul în care cunoaște adresa de rețea efectivă a sistemului. Într-un mediu UNIX, de exemplu, cineva poate utiliza `Telnet` pentru o adresă IP fizică aleatoare, cum ar fi 193.32.129.20, și să lucreze pe sistemul de la celălalt capăt (dacă există) ca și când ar sta lângă el, în afară de cazul în care se depun eforturi pentru a face adresa inaccesibilă.

Prima metodă logică de apărare pentru majoritatea siturilor ar fi probabil un *ruter*, un dispozitiv hardware care preia traficul de pe Web și îl retransmite în rețeaua locală ca și când acesta ar avea o origine locală, împiedicând în același timp trecerea anumitor tipuri de trafic. El realizează același lucru și pentru cererile care ies din rețea.

Deoarece adresele IP efective ale sistemelor individuale dintr-o rețea securizată nu sunt expuse niciodată, un nivel considerabil de securitate este inerent pentru fiecare ruter. Dacă un ruter are capacitatea de a filtra în mod selectiv pachetele, bazându-se pe origine sau pe portul destinație, atunci apare un nivel de securitate suplimentar. Majoritatea ruterelor comerciale actuale au această capacitate. Deci pentru a proteja presupusa înregistrare pacient, îi comunicați ruterului să permită numai anumite tipuri de acces controlat, `Telnet`-ul fiind refuzat în special, dar și `FTP`, `rlogin`, `rsh` și alți intruși periculoși.

Ruterele sunt omniprezente pe Web. De fapt, sunt personificarea hardware a lucrului în rețea și a Internetului în sine. Cu toate că ruterele oferă un nivel de securitate atunci când sunt setate să facă acest lucru, aproape fiecare informație văzută de dumneavoastră pe Internet a trecut printr-o serie de rutere, acestea fiind sistemele care trimit pachetele de date spre destinația lor finală.

Ele realizează acest lucru prin întreținerea unor tabele de rutare care listează căile la destinațiile cunoscute și adresa unui loc unde se trimit pachetele a căror destinație nu este cunoscută de ruter. Dacă un ruter adună trafic spre o anumită destinație, atunci el poate cere adresa de la un alt ruter și să o plaseze în tabela sa internă de rutare. Dacă ruterele cunosc modul în care să livreze un mesaj la o anumită adresă, atunci ele comunică acest lucru dacă sunt întrebate de cineva. Aceste două acțiuni sunt separate. Un ruter poate returna informații despre rutare fără a fi dispus să transmită traficul, sau poate transmite traficul fără să fie dispus să dezvăluie destinația finală.

Secțiunile următoare continuă studiul amănunțit al ruterelor, explicând pe scurt mecanismele de bază prin care traficul este transmis pe Internet.

Protocolul TCP/IP

Pentru a înțelege modul de funcționare al ruterelor, trebuie să știți câte ceva despre protocolul TCP/IP care face posibilă comunicarea pe Internet. Protocolul complet utilizează o serie de niveluri de protocoale idealizate, numită *stivă*. Figura 16.1 vă arată o reprezentare tipică a stivei protocolului TCP/IP. Singura comunicare efectivă are loc la nivelul fizic, după cum vă așteptați, dar mesajul care trece înainte și înapoi între diferitele niveluri dă aplicației impresia că ar comunica de fapt în mod direct cu o altă aplicație. Linia punctată reprezintă această impresie, cu toate că fiecare dintre protocoalele inferioare crede *de asemenea* că el comunică direct cu echivalentul său de pe celălalt sistem, deci Figura 16.1 ar putea conține linii punctate între toate nivelurile echivalente.

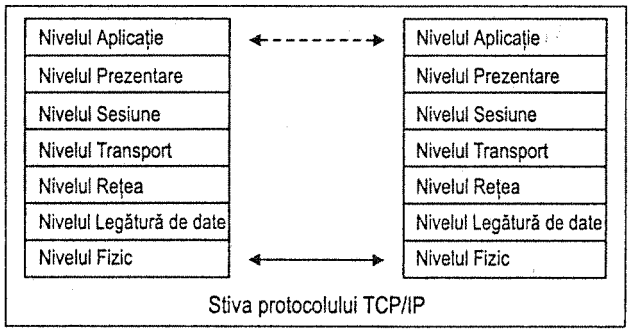


FIGURA 16.1
Stiva protocolului TCP/IP prezintă comunicarea între aplicații.

Aceste niveluri sunt mai degrabă reguli logice decât ținte fixe, și este un lucru destul de obișnuit să vedem niveluri combinate în anumite aplicații. TCP se află la nivelul Transport, iar IP la nivelul imediat inferior, nivelul Rețea. Cele două niveluri de jos, Legătură de date și Fizic, sunt implementate de obicei în firmware-ul care rulează pe placa dumneavoastră de rețea, deși pot fi implementate și în software. Nivelul Fizic realizează efectiv interfața cu hardware-ul pe care funcționează rețeaua dumneavoastră și manevrează operațiile de generare și descifrare a semnalelor electrice din fire sau a undelor electromagnetice din aer.

Prin participarea tuturor nivelurilor, TCP/IP împarte datele în segmente numite *pachete*, pe care le transmite pe Internet. Atunci când pachetele ajung la destinație ele sunt reasamblate în mesajul complet și livrate în final destinatarului. Acest proces este fragmentat în două părți: partea TCP se ocupă de dezasamblare, reasamblare și de corectarea erorilor, în timp ce IP se ocupă de adresarea și livrarea pachetelor la destinație.

IP – Internet Protocol (protocolul Internet)

Observați că IP doar trimite pachetele *spre* destinația lor finală. El utilizează un protocol *datagramă fără conexiuni*, care nu controlează în nici un fel dacă pachetele au ajuns sau nu. Deși pachetele au o adresă IP și un număr de port, itinerarul urmat de acele pachete pentru a ajunge la destinație este determinat euristic de către fiecare procesor intermediar. La destinație pot ajunge mai multe copii ale unui pachet sau nici una. Nu contează pentru IP.

Notă *Fără conexiuni* înseamnă că nu este păstrată o cale reală între origine și destinație. Calea urmată de un mesaj este determinată chiar de rețea, calea efectivă variind în funcție de modificarea condițiilor rețelei. O *datagramă* este pur și simplu un mesaj având adresa destinatarului. Ruterele trimit pachetele spre destinația lor, dar nu știu nimic despre conținutul acestora.

Nu există nici o metodă prin care IP să spună dacă un anumit pachet a ajuns sau nu, și nici măcar de unde a primit un anumit pachet. Acest lucru îl face vulnerabil la activitățile subversive ale oamenilor care doresc să obțină acces neautorizat la calculatoarele altora. Deși există în lucru implementări mult mai sigure ale IP, inclusiv versiuni care utilizează criptare și mecanisme de ne-respingere pentru a face atacurile mult mai dificile, numai o mică parte a traficului efectiv prin rețea utilizează aceste protocoale noi.

Deoarece în IP nu este încorporat nici un mecanism de corectare a erorilor și nu există nici o modalitate de a afla dacă un pachet a fost sau nu livrat, fiecare pachet este etichetat cu un câmp timp-de-viață care permite eliminarea sa din rețea după o perioadă care este specificată în acel câmp. De asemenea, ruterelor le este permis să elimine pachete pe care le pot manevra foarte greu sau pachete pe care nu vor să le transmită, iar acesta este modul în care se apără împotriva intruziunilor.

Un ruter care protejează sistemul dumneavoastră își anunță dorința de a ruta traficul spre sistemul dumneavoastră, fără a dezvălui adresa fizică efectivă. În sens contrar, atunci când un pachet de ieșire este trecut în coadă, ruterul anunță sistemele din rețeaua dumneavoastră că el reprezintă cea mai bună metodă ca pachetul să ajungă la destinație, fără a dezvălui nici o informație despre itinerare alternative. Deoarece orice ruter este liber să renunțe la orice pachet IP în orice moment, securitatea pe acesta poate fi asigurată numai dacă este mai pretențios decât de obicei față de pachetele pe care le trimite spre o destinație. Totul se realizează mai mult sau mai puțin automat și face parte din adevărata

inteligență a întregului protocol Internet. Ca întreg este foarte robust, în ciuda faptului că piesele intermediare pot fi slabe.

TCP – Transmission Control Protocol (protocolul de control al transmisiei)

Protocoalele cele mai sigure, cum sunt Virtual Private Network (VPN), Secure Sockets Layer (SSL) și Virtual Tunneling (Vtun), se bazează pe TCP care este responsabil pentru stabilirea unei conexiuni, împărțirea mesajelor în pachete pentru ca IP să le poată trimite ca datagrame, adunarea tuturor datagramelor sosite de la o anumită sesiune și recombinarea lor, precum și tratarea oricăror erori care apar pe parcurs.

TCP utilizează IP ca mecanism de transport, dar îi adaugă un număr limitat de caracteristici de securitate. Cel mai important lucru este numerotarea pachetelor, aceasta realizându-se prin incrementarea startului fiecărei sesiuni noi prin niște numere destul de mari și prin rezervarea pe loc a unui interval de mii de pachete. Toate pachetele care sosesc trebuie să aibă numărul de secvență corect, altfel sunt eliminate automat, considerându-se că nu aparțin aceleiași sesiuni. În plus, fiecare pachet de date care sosește trebuie confirmat. Deci dacă un pachet a fost interceptat, atunci procesul expeditor află acest lucru. Spoofing-ul (escrocarea) întregii tranzacții este destul de dificil pentru majoritatea oamenilor.

Deși acest lucru poate fi realizat folosind hardware costisitor, puțini oameni dispun de resursele necesare pentru a deturna (hijack) o sesiune. Cele mai obișnuite atacuri asupra securității se bazează pe subminarea comportamentului logic. Încercarea de a afla o parolă privind peste umărul cuiva sau prin deducerea ei logică este mult mai ușoară decât încercarea de a intercepta și descifra semnătura interferenței radio a anumitor acționări de taste.

De obicei, sistemele de informații medicale permit accesul pentru alți specialiști din domeniu, deseori printr-un ecran de login, deci multe sisteme medicale sunt vulnerabile la atacuri prin spoofing pentru preluarea unei sesiuni confidențiale aflate în derulare. Secțiunea „Spoofing" din acest capitol studiază mai amănunțit acest lucru.

O metodă obișnuită de atac logic este utilizarea diverselor „intrări secrete" încorporate în aproape toate sistemele de operare pentru a face posibile serviciile de utilizator. În ciuda faptului că un sit poate interzice cu desăvârșire încărcarea pe el a fișierelor de date de către utilizatori individuali, același sit permite unui serviciu e-mail să încarce mii de fișiere. Dacă se găsește o modalitate de a păcăli serviciul de e-mail astfel încât să creadă că fișierele dumneavoastră sunt de fapt fișiere de e-mail, atunci ați încărcat chiar acele fișiere pe care, teoretic, nu trebuia să puteți să le încărcați. În următoarele două secțiuni voi descrie în termeni generali două metode obișnuite de atac pentru a ilustra rafinamentul și mijloacele utilizate de cei care doresc să obțină acces neautorizat la sistemul dumneavoastră.

Internet Worm

Un exemplu vestit de atac prin Internet este „Internet Worm", creat de Robert Morris, un student în vârstă de 23 de ani de la Cornell, în 1988. Pe lângă diverse alte tehnici, „viermele" a utilizat serviciul de e-mail și un anumit program, lăsat deseori pornit pe sistemele UNIX, care transportă majoritatea traficului pe Internet.

Versiuni mai vechi ale acestui program, scrise în vremuri mai sigure, stocau cererile primite într-un buffer intern proiectat să fie suficient de mare pentru a trata orice cerere

legitimă. Viermele trimitea în mod deliberat un mesaj mai mare decât bufferul intern, care inunda bufferul și scria sfârșitul mesajului (care conținea cod executabil) în spațiul de cod al programului, unde acesta era executat imediat.

Prin exploatarea altor slăbiciuni grave în implementările mai vechi ale programului, viermele era capabil să caute utilizatorii de la distanță de pe alte sisteme utilizând aceeași tehnică, după care se autocopia pe acele sisteme și așa mai departe, într-un joc de programare de tip piramidă. Spre deosebire de schemele piramidale reale, în care făptașii rămân repede fără victime pe care să le exploateze, calculatoarele sunt destul de naive. Deci, lățimea de bandă Internet a fost repede umplută cu catralioane de copii ale viermelui care se răspândea peste tot. Aceasta a dus la o blocare aproape completă a traficului legitim pe Internet pentru mai multe zile, până când siturile au descoperit ce se întâmplă și s-au deconectat de la Internet pentru a preveni inundarea procesoarelor locale.

Partea tristă este că Morris pretinde că s-a gândit că va face ceva inteligent și a experimentat cu impertinență pe întreaga rețea, fără a cere acordul, presupunând că după ce a dovedit conceptele sale, va putea să se dezvăluie ca fiind geniul din spatele programului, pentru a primi laudele tuturor și oferte de serviciu tentante. Desigur, s-a străduit mult să ascundă sursa reală a viermelui încercând să își mascheze identitatea și rulându-l mai întâi de pe un calculator al unei alte universități, deci pretențiile sale de nevinovăție au fost destul de patetice pentru cei care l-au anchetat și judecat. În ciuda acestor fapte, a scăpat destul de ușor după standardele de azi, fiind pus în libertate și trebuind să plătească o amendă care nu prea avea legătură cu pagubele efective pe care le-a produs proprietarilor nenumăratelor sisteme pe care le-a zăpăcit. Oamenii care fac astăzi așa ceva ajung adesea la închisoare.

Există multe alte servicii Internet de care s-a abuzat în acest mod sau în moduri asemănătoare, deci urmăriți îndeaproape aplicațiile dumneavoastră în care aveți „încredere". Internet Worm ar fi putut fi proiectat la fel de ușor să caute înregistrări medicale importante și să le transmită prin e-mail unui cont anonim, de unde să fie ridicate ulterior de un indiscret.

Spoofing (escrocări)

Spoofing este un atac tehnic asupra securității unui sistem, care se bazează pe exploatarea unei relații de încredere existente. *Hijacking* (*deturnarea*) este un atac oarecum similar, dar care necesită mai multe cunoștințe și mult mai multe echipamente.

După cum s-a prezentat anterior în acest capitol, comunicațiile pe Internet au loc în principal prin protocolul TCP/IP, care permite sistemelor gazdă și client să comunice între ele printr-o imensă centrală telefonică comună. Deși mulți utilizatori compară conexiunea lor la Internet cu o conversație telefonică, cu un fir dedicat între stațiile lor de lucru și siturile Web pe care vor să le acceseze sau destinatarii de e-mail la care vor să ajungă, această concepție este eronată. În realitate, traficul pe Internet poate fi „pândit" de oricine are echipamentul tehnic potrivit și acces la o parte a fluxului de date.

Deoarece stiva protocolului TCP/IP este foarte robustă din proiectare, permițând direcționarea pachetelor IP în aproape orice loc, este destul de ușor să pretindeți că sunteți altcineva atâta timp cât puteți amuți într-un anumit mod calculatorul pe care îl întruchipați.

Nu voi intra în detalii complexe referitoare la modul de realizare a acestui lucru pentru a evita încurajarea experimentelor care ar putea duce la închisoare și dizgrație, dar acesta

este unul dintre cele mai obișnuite atacuri asupra sistemelor (serverelor) UNIX și ar trebui împiedicat de administratorul sitului dumneavoastră. Înainte de a încredința datele dumneavoastră unui server de pe Web, întrebați ce măsuri au fost luate pentru a preveni spoofing-ul sau deturnarea sesiunilor utilizatorilor. Administratorul dumneavoastră ar trebui cel puțin să includă ca parte a soluției analize periodice ale conectărilor la server. Comportamentul TCP poate fi și el modificat pentru a elimina predictibilitatea care ușurează atacurile spoofing.

Notă IP în noua sa variantă va include capacitatea de a cripta traficul, deci spargerea conexiunilor existente va deveni *puțin* mai grea. Totuși, criptarea nu este un înlocuitor al politicii complete de securitate, iar un administrator responsabil cu securitatea ar trebui să o considere doar un instrument din setul de instrumente necesare.

O altă soluție este un firewall sau un ruter care filtrează mesajele sosite ce par să provină *dintr-un* domeniu sigur, puțin probabil în viața reală. Un corolar la aceasta ar fi că pachetele care ies dintr-un domeniu sigur ar trebui filtrate pentru a elimina pachetele care par să provină din exteriorul său, lucru la fel de improbabil într-o derulare normală de evenimente.

Prima indică faptul că situl dumneavoastră este posibil să fie ținta atacului unui escroc, iar cea de a doua înseamnă că cineva de pe situl dumneavoastră încearcă să escrocheze un alt sit. Ambele sunt rele.

Închiderea intrărilor secrete de securitate

Închiderea intrărilor secrete ale sistemelor cu date sensibile este aproape tot ce pot face ruterele în privința securității. Prin ascunderea adreselor efective ale resurselor din rețea, totul trebuie efectuat prin control la distanță prin ruter. Deoarece ruterul poate fi setat astfel încât să refuze complet majoritatea tipurilor de control la distanță sau să permită accesul numai de la sisteme gazdă cunoscute, majoritatea tipurilor de escrocherii efectuate la intrare sunt limitate cu severitate. Majoritatea ruterelor și serverele din spatele lor păstrează jurnale. Deci dacă vedeți multe încercări de conectare eșuate, care au propriul lor port binecunoscut, sau alte accesări neautorizate, puteți fi destul de sigur că cineva încearcă să ghicească o parolă și puteți lua măsuri pentru a depista vinovatul înainte ca acesta să reușească. De multe ori, „spargerea" unui sistem sigur, care are un administrator de sistem vigilent, durează mai mult decât timpul de care are nevoie administratorul ca să depisteze identitatea hackerului, deci recidiviștii sunt deseori depistați. Atacul asupra unui calculator este considerat de cele mai multe ori delict federal, deci poate fi implicat și FBI-ul.

Notă Ca să nu deveniți prea alarmat, numărul efectiv de asalturi ale hackerilor asupra sistemelor de securitate a scăzut în ultimii ani. Scăderea se datorează îmbunătățirii securității, constrângerilor legale și faptului că mulți hackeri s-au maturizat suficient pentru a realiza că pot câștiga sume importante *lucrând* efectiv în lumea high-tech în loc să încurce lucrurile.

Ruterele nu sunt excesiv de scumpe, sisteme destul de bune fiind disponibile începând de la 400 $ (SUA). Cu cât aveți nevoie de o viteză mai mare și de mai multe conexiuni simultane, cu atât va crește și prețul ruterului. Ruterele comerciale disponibile ajung deja până la vitezele T1 și T3 (1.544.000 bps și, respectiv, 43.232.000 bps, în comparație cu un modem telefonic tipic, la 56.000 bps) și există mulți furnizori, unul dintre aceștia fiind Cisco Systems, la `http://www.cisco.com/`, care transportă destul de mult din traficul Internet și oferă o gamă de rutere potrivite pentru orice, de la cele pentru birouri până la furnizorii de backbone (coloană vertebrală) de mare viteză. Există și rutere capabile de viteze mult mai mari, până la OC192 (9,6 Gbps), utilizate între furnizorii importanți de rețele și institutele de cercetare.

Ruterele disponibile de obicei acoperă un interval de la sistemele mici, potrivite pentru birourile individuale sau ale micilor companii, cum sunt cele disponibile de la Cisco Cayman Sistems, la `http://www.cayman.com/`, sau de la Netopia, la `http://www.netopia.com/`, și până la cele potrivite pentru backbone-urile Internet importante, cum sunt cele de la Juniper Networks, la `http://www.juniper.net/`.

Construirea unui firewall de securitate

Un *firewall* este o barieră, formată din software și hardware, între lumea exterioară și rețeaua sau calculatorul dumneavoastră. El urmărește atacurile asupra unui sistem utilizând o varietate de tehnici complexe, previne atacurile atunci când poate, ține un jurnal despre atacuri pentru ca dumneavoastră să știți când a fost atacat sistemul dumneavoastră și poate chiar să vă anunțe în timp real. Diferențele dintre rutere și firewall-uri s-au estompat în ultimii ani, multe rutere oferind un anumit nivel de funcționalitate firewall și multe firewall-uri comportându-se ca niște rutere virtuale.

Presupusa înregistrare medicală este o țintă tentantă, deoarece acest gen de informații reprezintă de fapt un articol de comerț în anumite cercuri, deci trebuie să închideți toate punctele de intrare posibile, în timp ce urmăriți încercările de pătrundere. În cazul în care asemănați datele dumneavoastră cu banii dintr-o bancă, un firewall este ca un seif permițând un acces controlat unui grup restrâns, ridicând în același timp bariere puternice pentru orice altă persoană. Dar chiar și cele mai puternice seifuri au nevoie de camere video de supraveghere pentru a depista o încercare de intrare neautorizată, iar jurnalele sistemelor sunt echivalentul computerizat al camerelor video de securitate ale unei bănci.

Utilizarea unui firewall și/sau a unui ruter poate fi vitală, în special atunci când utilizați camere de conversație on-line, mesagerie ICQ dezordonată și altele asemenea, care sunt frecventate de obicei de persoane cu mult timp liber. Unele dintre aceste persoane pot fi capabile să spioneze hard-discul dumneavoastră în timp ce conversați. Modemurile prin cablu dirijează traficul rețelei dumneavoastră pe la toată lumea care este pe același cablu. Toți stau acasă fără să fie supravegheați de nimeni. Este absolut necesară o anumită protecție.

Ca model, un firewall este un singur calculator cu două plăci de rețea, una orientată spre Internet, iar cealaltă spre o rețea particulară sigură. Sistemul firewall examinează adresele sursă și destinație, numărul de port conținut în antetul fiecărui pachet primit, și filtrează pachetul în funcție de ceea ce găsește.

Deoarece majoritatea serviciilor de rețea ascultă și emit la porturi binecunoscute, acest lucru are ca rezultat prevenirea completă a anumitor tipuri de cereri de servicii. De exemplu, din motive de securitate, `finger` la portul 79 este blocat deseori de administrator pentru traficul care intră. `finger` arată dacă un utilizator este sau nu logat (conectat),

lucru care poate fi nesigur dacă îl știe oricine. Prin împiedicarea serviciului `finger`, se previne situația în care cineva din exterior încearcă să determine dacă o femeie lucrează până târziu la birou, riscând astfel să fie pândită atunci când pleacă.

`telnet` la portul 23 și `ftp` la porturile 20 și 21 sunt de asemenea făcute indisponibile de către administratorii de securitate zeloși. Acestea pot fi ocolite utilizând formulare HTTP de încărcare, și altele asemenea, deși este mai puțin comod pentru utilizator, dar prezintă riscuri grave atunci când pot fi accesate liber de oricine.

Utilizarea serverelor proxy

Un *server proxy* este într-un fel opusul unui firewall, deoarece este proiectat în primul rând pentru a păstra utilizatorii în interior. Un bun efect secundar de securitate pe care îl are este acela că permite utilizatorilor din rețeaua dumneavoastră internă să acceseze Internetul într-un mod controlat și îi ascunde atunci când ei doresc acest lucru. Un proxy realizează acest lucru prin monitorizarea și interceptarea cererilor utilizatorilor și reformularea cererilor utilizând propria sa adresă de rețea. Atunci când primește răspunsul de la rețea, proxy-ul transferă datele utilizatorului inițial.

Deoarece lumea din exterior nu vede niciodată adresele din rețeaua internă, este mai greu pentru o persoană răuvoitoare din exterior să pătrundă în interiorul unui firewall bine proiectat, chiar dacă acea persoană are acces la HTTP și la jurnalele serverului și poate cerceta cererile sosite. Pe de altă parte, un utilizator inteligent din interior poate utiliza prin proxy orice serviciu direct permis, putând face aproape orice dorește, deci atât firewall-urile cât și serverele proxy necesită o rezervă de siguranță puternică, sub forma unui administrator de sisteme de securitate energic și meticulos.

Un instrument de bază pentru administrarea sistemelor de securitate este monitorizarea atentă și continuă a jurnalelor create de firewall și de proxy pentru a detecta atât încercările incorecte de accesare din exterior a rețelei interne, cât și încercările incorecte de a evita detectarea de către serverul proxy. Răufăcătorii din interior care acționează în afară sunt la fel de periculoși ca și cei din exterior care încearcă să pătrundă înăuntru și este mai probabil să fie nemulțumiți și să cunoască mai multe despre securitatea internă.

De asemenea, cercetarea jurnalelor de pe proxy permite organizațiilor să monitorizeze întreaga activitate din rețea a angajaților lor, deci aceste jurnale sunt utilizate de multe ori pentru a descoperi și a pedepsi angajații care încalcă politica organizației cu privire la utilizarea Internetului.

Proxy-urile implementează deseori și un cache intern, deci cererile obișnuite către Internet pot fi scurtcircuitate prin furnizarea directă la utilizator a informațiilor găsite în cache, o economie importantă de timp și resurse.

Porturi binecunoscute și alte servicii sistem

O cale de a proteja sistemul dumneavoastră de cereri de servicii neautorizate este cunoașterea portului la care poate sosi cererea. Secțiunea următoare furnizează o listă de servicii obișnuite.

Alocările porturilor binecunoscute

Porturile binecunoscute au fost utilizate încă din primele zile și sunt „binecunoscute" deoarece există datorită unei înțelegeri comune și sunt standardizate pentru întregul

Internet. Porturile binecunoscute se află în intervalul 0–1023, deși toate alocările foarte vechi sunt sub 255. RFC-1700 Assigned Numbers (numere atribuite) descrie porturile binecunoscute actuale, dar poate fi reformulată oricând cu alte numere pentru actualizarea listei. Acest lucru este prezentat în Tabelul 16.1.

Tabelul 16.1 Cele mai utilizate numere de porturi de pe Internet

Numele serviciului	Port	Descrierea serviciului
	0	Rezervat
`ftp-data`	20	Transfer de fișiere (date prestabilite)
`ftp`	21	Transfer de fișiere (control)
`ssh`	22	Protocol SSH de login la distanță
`telnet`	23	Telnet
`smtp`	25	Transfer simplu de mesaje poștale
`time`	37	Timp
`nameserver`	42	Server nume de gazdă
`nicname`	43	whois (date referitoare la utilizatorii înregistrați)
`domain`	53	DNS (server cu nume de domeniu)
`gopher`	70	Gopher
`finger`	79	Finger
`http`	80	HTTP World Wide Web
`kerberos`	88	Kerberos
`pop2`	109	Post Office Protocol (protocolul poștal) – versiunea 2
`pop3`	110	Post Office Protocol – versiunea 3
`nntp`	119	Network News Transfer Protocol (protocolul de transfer de informații în rețea)
`ntp`	123	Network Time Protocol (protocolul de timp în rețea)
`snmp`	161	SNMP (protocolul de gestionare a rețelei)
`snmptrap`	162	SNMPTRAP
`send`	169	SEND
`irc`	194	Protocolul Internet Relay Chat
`https`	443	Protocol HTTP peste TLS/SSL
`ph`	481	Serviciul ph
`uucp`	540	uucpd
`uucp-rlogin`	541	uucp-rlogin
`cybercash`	551	CyberCash
`doom`	666	Doom id Software

Tabelul anterior conține doar câteva dintre porturile binecunoscute, deoarece majoritatea porturilor binecunoscute sunt pentru protocoale și pentru aplicații software despre care aproape nimeni nu a auzit vreodată. Jocul Doom, evident, are propriul număr de port pentru a facilita sesiunile de joc în rețea pentru programatori și pentru cei cu mult timp liber. Quake, Unreal și Kali au parte de aceeași binecuvântare.

Alocări de porturi înregistrate

Porturile înregistrate sunt în principal cele cerute de producători și furnizori pentru aplicațiile lor particulare, deși câteva sunt utilizate de HTTP și de alte servicii pentru rețele. Este un lucru obișnuit ca programatorii să își atribuie propriile numere de porturi în acest interval. Porturile înregistrate se situează în intervalul 1024–49151.

Tabelul 16.2 Porturi înregistrate

Numele serviciului	Port	Descrierea serviciului
	1024	Rezervat
socks	1080	Sock
http-alt	8008	HTTP alternativ
http-alt	8080	HTTP alternativ (vezi portul 80)

Aici sunt trecute doar câteva dintre ele, deoarece majoritatea sunt utilizate de către aplicații brevetate și de către servicii cu care majoritatea oamenilor nu se întâlnesc niciodată.

Alocarea dinamică a porturilor

Alocările dinamice de porturi sunt utilizate, printre altele, pentru fluxurile audio și video sau pentru orice serviciu care poate necesita un port dedicat pentru orice interval de timp. Multe aplicații permit reglarea cerințelor porturilor lor dinamice, deci intervalul imens și, teoretic, imposibil de controlat poate fi redus considerabil atunci când este nevoie să se treacă prin servere proxy sau firewall-uri. Singurul caz în care trebuie să aveți grijă atunci când lucrați cu ele este când setați serverul dumneavoastră proxy sau firewall, deoarece va trebui ca ele să fie configurate ca fiind disponibile pentru utilizare în sisteme sigure.

Porturile dinamice și/sau private se află în intervalul 49152–65535. Acestea nu sunt trecute aici, deoarece pot avea orice utilizare specială pe care doriți să le-o atribuiți.

HTTP (HyperText Transfer Protocol) – Protocolul de transfer hipertext

Porturile sunt utilizate adesea de către o aplicație pe o bază prestabilită, deci dacă nu se specifică altceva, o cerere HTTP a browserului dumneavoastră pleacă spre portul 80. De asemenea, acestea pot fi introduse direct prin utilizarea întregului protocol de adresare HTTP, care vă permite să introduceți un număr de port după adresa HTTP a unui sit, cum ar fi `http://www.oricenume.com:numarport/`, unde *numarport* poate fi orice număr de port valid ascultat de un proces server.

De obicei, o versiune sigură a HTTP poate fi la un număr de port diferit de un HTTP nesigur – adesea 8080 sau 8008, cu toate că 443 este la fel de obișnuit – deci legăturile interne sigure pot indica la `http://www.oricenume.com:8080/paginasigura.html`.

În schimb, serverul poate fi setat să trateze `https://www.oricenume.com/` altfel decât pe `http://www.oricenume.com/` și să se ocupe el de problemele de rutare prin port. Serverele sunt de fapt destul de inteligente și pot realiza multe lucruri magice fără știrea dumneavoastră. Unul dintre lucrurile la care se pricep foarte bine este tratarea unor cereri în moduri speciale, bazându-se pe cele câteva informații pe care browserul le trimite împreună cu fiecare cerere.

Sugestie de la LeeAnne.com Words to weave by

Simplul fapt că o pagină *pare* simplă nu înseamnă că *este* simplă. Logo-ul W3C de pe un sit se numește de fapt `W3C.html` în codul sursă al paginii. Fiecare cerere pentru fișierul `W3C.html` este tradusă automat într-o formă potrivită, text sau imagine, pentru browserul utilizat la accesarea paginii de bază.

Mulți proiectanți Web care utilizează o redirectare bazată pe HTML și o reconfigurare dinamică bazată pe identificarea browserului pot folosi probabil mai bine caracteristicile bazate pe server pentru a realiza același lucru într-un mod mult mai demn de încredere și cu mult mai puține efecte secundare.

Produse firewall existente

În lumea serverelor, nume vestite de firewall-uri, cum sunt Check Point, la `http://www.checkpoint.com/`, Trusted Information Systems (acum Network Associates), la `http://www.nai.com/`, sau Raptor Systems (acum AXENT), la `http://www.axent.com/`, sunt scumpe, dar foarte potrivite pentru medii comerciale de mari dimensiuni.

Totuși, firewall-urile nu intră numai în preocupările siturilor marilor corporații. Pe măsură ce tot mai multe persoane și mici companii sunt conectate la Internet prin adrese IP dedicate, cum ar fi DSL sau conexiunile prin cablu, toate devin vulnerabile la aceleași tipuri de atacuri care sunt lansate asupra siturilor guvernamentale sau ale corporațiilor. Deși pot fi lansate atacuri aleatoare la orice adresă IP, siturile individuale și ale micilor companii sunt de obicei mai puțin protejate decât cele ale corporațiilor. Dacă aveți un calculator acasă, este acesta protejat prin parolă? Parola este *bună*? O modificați la intervale regulate? Aveți *vreo metodă* de depistare a faptului că cineva încearcă să pătrundă în sistemul dumneavoastră? Aveți un ruter în locul unui modem?

Pentru cei mai mulți dintre noi, răspunsul este nu. Iar ISP-urile, rețelele de televiziune prin cablu și companiile telefonice care vând acces la Internet nu sunt prea preocupate de riscurile reale ale conectării on-line. Totuși, există câteva produse de securitate proiectate pentru companiile mici și pentru utilizatorii casnici, cum ar fi ConSeal Private Desktop de la Signal 9 Solutions, la `http://www.signal9.com/`, și BlackICE Defender de la Network ICE, la `http://www.networkice.com/`. AtGuard de la WRQ (`http://www.wrq.com/`) a fost încorporat în Norton Internet Security 2000 de la Symantec, la `http://www.symantec.com/`, care mai conține control parental și software antivirus într-un singur pachet. Prețul pentru majoritatea acestor produse este în jurul sumei de 50$ (SUA).

Figura 16.2 prezintă ecranul de control pentru PC Firewall de la Signal 9, care conține atât aptitudinea de a schimba mesaje criptate (fără cost suplimentar), cât și protecție împotriva majorității atacurilor prin rețea. Sunt prezentate un număr mare de reguli și pot fi adăugate multe altele, dacă situația o cere. Elementele din jurnal nu sunt de fapt atacuri, ci încercări ale ICQ de a anunța prezența mea pe Internet. Aceasta este o fisură importantă în privința securității, deoarece ICQ și alte pachete asemănătoare lasă pe

oricine să afle care este adresa dumneavoastră de IP chiar în acest moment și, de fapt, ICQ a fost utilizat pentru a lansa atacuri asupra calculatoarelor altor persoane. Actualizați întotdeauna aceste programe, deoarece găuri în securitate sunt găsite și reparate periodic. Sau chiar mai bine, gândiți-vă să nu le utilizați deloc, sau să restrângeți la un mic grup de prieteni numărul persoanelor care pot accesa informația dumneavoastră ICQ. Signal 9 are disponibil, de asemenea, și software-ul Virtual Private Network (VPN) care setează căi de comunicație criptate pe Internet, permițându-vă să considerați Internetul ca pe o rețea locală sigură și să vă legați la birouri și situri aflate la distanță ca și când acestea s-ar afla în încăperea alăturată.

Notă Aplicațiile ușor de utilizat, cum sunt Back Orifice de la cDc, L0phtCrack de la L0pht Heavy Industries și NetBus de la Neikter le permit și unor idioți să „spargă" sistemul dumneavoastră neprotejat așa cum o fac cei inteligenți. Nu trebuie să aveți cunoștințe speciale pentru a „sparge" un sistem din zilele noastre, ci numai un mic program comod pe hard-disc. Și aici nici nu luăm în considerare Satan, presupus ca fiind un instrument al administratorilor de sistem pentru descoperirea slăbiciunilor de securitate în propriul sistem, dar utilizat deseori și de către hackeri pentru a se infiltra în sistemele care nu le aparțin.

Evident, acestea nu sunt singurele firewall-uri disponibile pentru calculatoarele de acasă și din companiile mici, dar sunt două dintre cele mai bune și mai larg disponibile din punct de vedere comercial. O mică cercetare pe Web va dezvălui mult mai multe.

Atenție Dacă publicați adresa dumneavoastră de IP, este ca și cum ați anunța oamenii în spatele cărui tablou este ascuns seiful dumneavoastră. Dacă nu sunteți foarte sigur că seiful în sine rezistă la un atac hotărât, atunci ar fi mai bine să tăinuiți existența sa. Dacă aveți un sit Web care trebuie să își dezvăluie adresa IP, atunci țineți-l pe un sistem și într-o rețea separată de viața dumneavoastră personală sau de afaceri și faceți foarte dificil accesul de la el la rețeaua dumneavoastră internă. Multe organizații impun ca descărcările pe serverul Web să fie făcute pe dischete sau pe alte medii fizice de transport.

Aceste sisteme „firewall" simple sunt numite astfel prin analogie. Un firewall adevărat este un software asistat de hardware, interfețe de rețea care sunt conectate fizic la alte rețele. Prin extindere, vă puteți gândi la un firewall ca la un proces similar care stă pe un singur sistem și filtrează pachetele înainte de a le permite să acceseze sistemul propriu-zis, deși nu există o diferență reală între hardware-ul de acces al rețelei și sistemul vulnerabil.

Atunci când sosește o cerere de pe Internet, firewall-ul decide dacă o va lăsa să treacă sau nu. De obicei, firewall-urile permit trecerea cererilor HTTP, deși nu în mod obligatoriu, precum și a altor tipuri de informații definite clar. Orice lucru periculos este eliminat.

Cea mai bună măsură este să nu permiteți nici o intrare și nici o ieșire. Aceasta este metoda de securitate utilizată de cele mai sigure calculatoare din domeniul militar. Ele sunt înconjurate de o colivie de sârmă și cablurile de alimentare sunt singurele conectate la ele. Toate informațiile sunt introduse și extrase pe hârtie sau pe alte medii de transfer sigure. Hârtia și celelalte medii sunt tocate cu regularitate cu dispozitive scumpe care transformă documentele în terci. Inventarea sistemelor de securitate nu prea este de folos dacă aruncați la gunoi toate secretele dumneavoastră ca să le poată citi oricine. O

abordare ușor diferită a firewall-urilor este utilizată de aplicația Private Desktop de la aceeași companie Signal 9 și este prezentată în Figura 16.3. Acest pachet monitorizează de fapt traficul care iese din sistem după numele programului care l-a generat, deci chiar dacă un „vierme" a reușit să se infiltreze în sistemul dumneavoastră fără să fie detectat, aptitudinea sa de a comunica în exterior va fi retezată cu asprime. Acest pachet este destinat cu precădere utilizatorilor casnici și acelor locuri în care PC Firewall necesită un anumit nivel de înțelegere a protocoalelor de bază pentru a determina ce înseamnă o anumită regulă. Private Desktop de la Signal 9 ascunde de utilizator informațiile de care nu este nevoie și permite să treacă aproape întregul trafic după o interogare pe ecran simplă și ușor de înțeles.

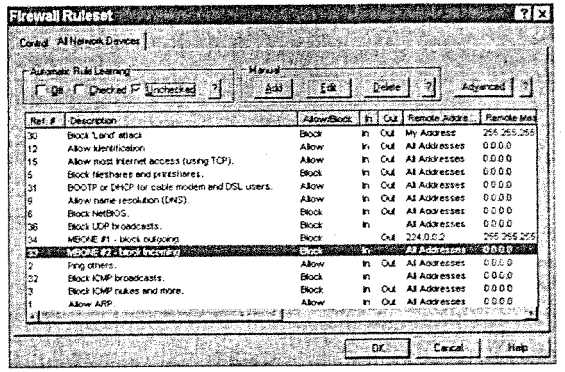


FIGURA 16.2
Ecranul de control din PC Firewall de la Signal 9 prezentând regulile utilizate.

Ecranul Signal 9 a fost generat la prima utilizare a programelor care necesită o conectivitate la rețea pentru a funcționa.

Un alt produs firewall pentru acasă și pentru companiile mici este BlackICE Defender de la Network ICE. El are caracteristicile necesare pentru a împiedica intruziuni și, în plus, conține o caracteristică unică, ICEcap, un program însoțitor care adună și coordonează informațiile referitoare la încercările de intruziune din întreaga rețea, deci pentru mediile mai mari, raportarea și analiza pot fi făcute comod într-o singură locație. Figura 16.4 prezintă ecranul de analiză ICEcap.

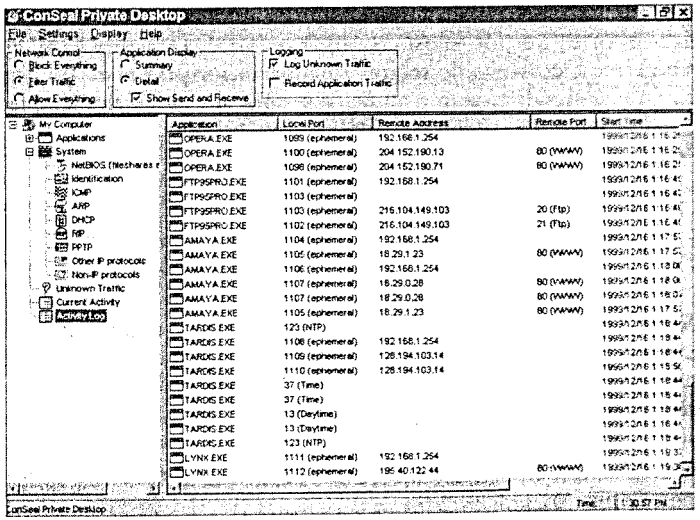


FIGURA 16.3
Ecranul de control din ConSeal Private Desktop de la Signal 9 prezintă jurnalul de activitate.

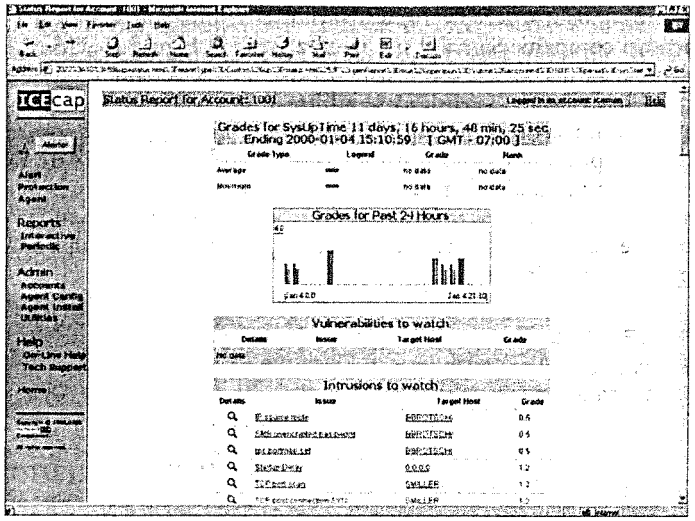


FIGURA 16.4
Produsul ICEcap de la Network ICE prezintă datele adunate de la mai multe sisteme protejate de BlackICE.

Crearea de parole unice

Fantezia hollywoodiană în care un copil isteț dă telefon la o bază de arme nucleare din vecinătate și pornește al treilea război mondial spunând ca parolă numele câinelui președintelui este doar atât, o fantezie. Dar cei mai mulți dintre noi, oamenii obișnuiți, folosim parole de acces similare cu cea a președintelui din film. Folosim numele sau prenumele soțului sau numele de fată al soției (o alegere foarte populară), numele filmului preferat sau al unui personaj dintr-un joc (metodă foarte populară printre tinerii necăsătoriți), o dată de naștere, numele animalului de companie, un model oarecare format pe tastatură, sau ceva de acest fel. Multe instrumente de hacking se bazează pe această slăbiciune umană, deci o parte importantă a unui sistem de securitate este convingerea utilizatorilor că nu pot folosi ca parolă un șir de caractere banal și ușor de ghicit. Multe situri trec încercările de schimbare a parolelor printr-un filtru care împiedică folosirea cuvintelor din dicționar, a numelor proprii și a modelelor simple de pe tastatură.

Atenție Fiți atent dacă folosiți Satan și alte instrumente de hacker pe un sistem care nu vă aparține, deoarece este o metodă excelentă de a fi dat afară din serviciu, eliminat de ISP-ul dumneavoastră, trimis la închisoare, falimentat și divorțat sau izolat de societate, nu neapărat în această ordine. Oamenii care se ocupă de securitatea calculatoarelor nu se amuză ușor și adesea înțeleg destul de greu o glumă.

Majoritatea sistemelor de acasă sau de la serviciu sunt extrem de vulnerabile. Fără o modalitate de a detecta și de a înregistra într-un jurnal atacurile la securitate, un hacker poate încerca secvențial o listă de nume feminine comune, nume de personaje (ați fi surprins dacă ați ști câți se cred JamesBond sau Agent007) și cuvinte de dicționar până când ceva se potrivește. Această operație poate fi făcută automat, fără intervenție umană, cu ajutorul unui demon. În ciuda părerii generale, *DukeNukem* nu este o parolă extrem de inteligentă.

O metodă de a crea o parolă ușor de memorat este să folosiți primul sau un alt caracter din fiecare cuvânt al unei fraze unice cu o singură modificare. „Pipăruș Petru *8* a plecat cu papucii proprii" va deveni „PP8apcpp" (însă menționarea acestei parole aici o face extrem de cunoscută).

Pentru o securitate mai bună și pentru o parolă mai lungă, încercați o frază mai lungă, cum ar fi „Sunt o fată drăguță din Olanda, cea mai drăguță dintre toate, iar toți băieții din echipa de baseball sunt nebuni după mine". Dacă luați prima literă din fiecare cuvânt, rezultatul va fi „SofddO,cmddt,itbdedbsndm", destul de greu de ghicit de cineva din afară. Fraza aceasta este una pe care eu personal nu o pot uita, deoarece am repetat-o aproape zilnic în copilăria mea, în timp ce săream coarda. Ar fi și mai bună dacă aș schimba-o pe ici pe colo, dar ați înțeles ideea. Desigur, dacă modificați fraza de bază, asigurați-vă că va *rămâne* ușor de memorat chiar după ajustare.

Sugestie de la LeeAnne.com Words to weave by — Chiar mai bună decât o parolă este o schemă de identificare fizică fără parole refolosibile. De fiecare dată când vă logați pe un sistem prin rețea, există o probabilitate semnificativă ca parola să fie depistată. Există multe sisteme de identificare, de la sisteme de acces cu carduri până la dispozitive de scanat retina.

Puteți folosi aproape orice există deja în memoria dumneavoastră, deoarece faptul că se află acolo presupune că ați trecut deja de partea cea mai grea și vă puteți concentra pe algoritmul folosit pentru a extrage literele. Frazele personale sunt mai bune decât cele comune: „Numele de fată al bunicii era Mathilde Pedersen" („NdfabeMP") este evident mai bună decât „Cine se scoală de dimineață, departe ajunge" („Cssdd,da") deoarece mult mai puține persoane pot ghici prima frază, chiar dacă ambele parole au opt caractere. Nu folosiți niciodată o parolă mai mică de șase sau opt caractere. Este prea ușor să se cicleze prin toate combinațiile de caractere posibile dacă folosiți o parolă scurtă.

Utilizați propria dumneavoastră frază lungă. Folosiți ultimele litere. Ajustați fraza. Spuneți-o de la coadă la cap. Fiți inventiv. Dar, în special, nu folosiți „Batman!" sau ceva asemănător.

Sugestie de la LeeAnne.com Words to weave by — O parolă bună *trebuie* să fie ușor de reținut, astfel încât să nu trebuiască să o scrieți pe etichete autocolante. De asemenea, trebuie să fie greu de ghicit. Amestecați numere, semne de punctuație și simboluri, nu numai litere. Și trebuie să vă schimbați parola în mod regulat și de fiecare dată după ce ați fost nevoit să o introduceți de la o locație aflată la distanță.

Criptare

Criptarea este procesul de amestecare a datelor astfel încât acestea să poată fi citite numai de cei care au informații secrete despre modul în care au fost amestecate datele. Știința *criptologiei* (studiul algoritmilor de criptare) a parcurs un drum lung de la păsărească pe care o foloseam când eram copii. Termenul mai vechi pentru această știință misterioasă era *cifrare*, de la cuvântul arab pentru zero, *sifr*, care era atât de misterios în acele vremuri încât părea magic. Astfel, trasformarea unei harababuri de litere într-un text simplu părea că face să apară ceva din nimic, iar descifrarea cifrurilor a devenit un secret de stat încredințat autorităților militare și civile.

Notă Primul program de calculator stocat electronic, Colossus, a fost dezvoltat de Alan Turing şi asociaţii săi, sub auspiciile guvernului britanic, pentru a sparge codul Enigma al germanilor. Întregul proiect era atât de secret, încât au permis grupului ENIAC să-şi clameze prioritatea mulţi ani la rând. De fapt, mitul ENIAC este atât de vestit încât şi acum este descris în acest mod.

Schemele moderne de criptare de securitate maximă pot rezista la majoritatea asalturilor asupra informaţiilor din sistem, chiar dacă nu protejează integritatea sistemului. Astfel, chiar dacă un hacker reuşeşte să se strecoare printr-o intrare secretă sau să vă intercepteze mesajele e-mail, dacă fişierele de date şi mesajele sunt criptate şi semnate digital, atunci probabilitatea de a putea face ceva, în afară de a şterge fişierele şi a deranja lucrurile, este mult redusă.

Există două tipuri principale de tehnici de criptare: *simetrice* şi *asimetrice*. În cazul tehnicilor simetrice, aceeaşi parolă este folosită atât la criptarea, cât şi la decriptarea mesajului. În sistemele asimetrice, o cheie este folosită pentru a cripta mesajul, iar alta pentru a-l decripta. Este posibilă şi folosirea unei combinaţii a ambelor metode.

Coduri simetrice

Primele coduri au fost simetrice deoarece nu exista puterea de calcul necesară pentru a estima funcţii matematice care nu sunt inversabile, indiferent de complexitatea acestora. Cifrul clasic Cezar este un exemplu trivial, având o cheie de o singură literă. Mesajul se codifică înlocuind fiecare literă cu o alta, astfel: în locul lui A se pune cheia aleasă, iar celelalte litere sunt înlocuite prin corespondentele lor aflate la aceeaşi distanţă de ele ca şi A faţă de cheie. Decodarea se face prin inversarea procesului. Se spune că Cezar folosea cheia C (vanitate?):

```
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS
VJG SWKEM DTQYP HQZ LWORGF QXGT VJG NCBA FQIU
```

Într-o eră în care majoritatea oamenilor erau analfabeţi, această metodă era destul de sigură.

Cifrul Cezar se spune că a fost inventat de Iulius Cezar, lucru evident neadevărat, deoarece acest tip de cifru era folosit cu mii de ani înaintea sa şi a fost utilizat chiar şi în Egiptul antic. Dar cifrul Cezar *a fost* evident utilizat de legiunile romane, pentru comunicaţii secrete. Este rapid şi uşor de ţinut minte, deci nu trebuie să păstraţi însemnări care, găsite de spioni, ar trăda cifrul.

De atunci, cifrurile simetrice au evoluat şi sunt de nerecunoscut, devenind chiar o ramură a matematicii şi informaticii. Dar toate cifrurile simetrice necesită o comunicare între părţi înainte de a fi asigurată securitatea. Adică, se transportă într-un anumit mod fie cheia, fie informaţii despre alcătuirea cheii şi, în plus, această informaţie trebuie păstrată undeva, deci aceste sisteme sunt vulnerabile fie la un atac fizic, fie pur şi simplu faţă de oricine ar trage cu urechea.

Coduri asimetrice

Schemele de criptare asimetrice rezolvă problema comunicării cheilor între partenerii unei comunicaţii criptografice prin dezvăluirea în mod public a unei chei, lucru care permite oricui să trimită un mesaj sigur oricărui destinatar, folosind o cheie publică. Ele funcţionează prin intermediul unor funcţii trap-door care nu sunt inversabile. Acestea sunt operaţii matematice care pot fi efectuate uşor, dar a căror dezlegare este mult mai

complicată fără o cheie secretă. Deci, destinatarul mesajului criptat îl decriptează cu cheia sa secretă, iar textul simplu va fi dezvăluit.

Deţinătorul unei chei private poate cripta de asemenea un mesaj care va putea fi citit de oricine are acces la cheia sa publică, lucru care nu se prea potriveşte pentru transferul de secrete; totuşi, alcătuieşte baza semnăturilor digitale despre care veţi învăţa mai multe în secţiunea „Semnături digitale" din acest capitol.

PGP este unul dintre multele sisteme, după cum s-a menţionat anterior, care pot fi experimentate uşor şi ieftin, dar merită o menţiune specială. Iată unul din diversele situri pe care PGP este distribuit gratuit: `http://web.mit.edu/network/pgp.html`.

Cifrurile asimetrice vă permit să faceţi publică cheia dumneavoastră de criptare, permiţând oricui să vă trimită mesaje criptate şi împiedicând orice altă persoană, în afară de dumneavoastră, să decripteze mesajul cifrat.

Cealaltă parte a problemei este că puteţi semna un mesaj utilizând cheia dumneavoastră privată şi oricine cunoaşte cheia dumneavoastră publică îl poate descifra şi verifica dacă mesajul provine într-adevăr de la dumneavoastră. În cele din urmă, acest lucru va permite ca documentele electronice să fie semnate electronic, iar tranzacţiile on-line să poată fi folosite ca probe în procesele intentate. Mai multe ţări europene recunosc deja semnăturile electronice, iar Statele Unite vor ajunge la acest stadiu în cele din urmă.

Fragmentul extraordinar de confuz prezentat în Listingul 16.1 este propria mea cheie publică electronică, foarte sigură (pe 4096 de biţi), care foloseşte PGP pentru ID-ul meu de e-mail prestabilit.

Listingul 16.1 Cheia PGP a autorului

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGP for Personal Privacy 5.5.2

mQGiBDhaiyERBADcSohdD9yGzWIP1QZ9aZImDN9drNyMImxsOmKdPOWmU5VuM7sZ
ptrFEWonNsTFQmBbtYVQztLpztaapdLiRO5vJHv/qnA/150EN3eojUCh6qMmhu3y
Wh/cDqr9N+xC8YI+VoP+/ix4zA+jI9GAHCydJRnvP6opDHFkOYt9IE3BVwCg/7xp
LUEOUOJeGGtq2N6q7qERUuUEAK3ORmTBWS9XvYxNSN1/OZTpKOnZ3tyCOz378Evs
WMriY2y1ngJj1yeFi5a3GJ7KseIsgQUHxLB+unY5X/T9nR7PJGRisOfnjN+sACOj
W/ik/EXK1G8Y7t/DBjDwlcatj92uCVpEzlfBSFCOFjuglwSQuGQOAUT6xh4qhWEi
eUx9BADH3gmnODZi8TfHFvPkKeHzxNhLsGY+Os96GbhrzE51EwY7avrbgFLYe4Z3
Yl6AoZzQOcUFBS1U1PYSiOGKse2+JDFbGHcNyKs8FK16tyKy+cwCkVHnbBtDQWtk
LtDFHs9acOcgX3d/Qzv+IQOpffgVOjUcxaniixzODqa3xtKI7bQnTGVlIEFubmUg
UGhpbGxpcHMgPGxlZWFubmVAbGVlYW5uZS5jb20+iQBLBBARAgALBQI4WoshBAsD
AgEACgkQ+e69xICJydpOrQCg7z178bs3JFLNmd5esCRO1tnRXccAoPaCFE7TjV+g
h2AMnbILSAl1AHNxuQQNBDhaiyIQEAD5GKB+WgZhekOQldwFbIeG7GHszUUfDtjg
o3nGydx6C6zkP+NGlLYwSlPXfAIWSIC1FeUpmamfB3TT/+OhxZYgTphluNgN7hBd
q7YXHFHYUMoiVOMpvpXoVis4eFwL2/hMTdXjqkbM+84X6CqdFGHjhKlPOYOEqHm2
74+nQOYIxswdd1ckOErixPDojhNnl06SE2H22+slDhf99pj3yHx5sHIdOHX79sFz
xIMRJitDYMPj6NYK/aEoJguuqa6zZQ+iAFMBoHzWq6MSHvoPKs4fdIRPyvMX86RA
6dfSd7ZCLQI2wSbLaF6dfJgJCo1+Le3kXXn11JJPmxiO/CqnS3wy9kJXtwh/CBdy
orrWqULzBej5UxE5T7bxbrlLOCDaAadWoxTpjOBV89AHxstDqZSt90xkhkn4DIO9
ZekX1KHTUPj1WV/cdlJPPT2N286Z4VeSWc39uK50T8X8dryDxUcwYc58yWb/Ffm7
/ZFexwGqO1uejaClcjrUGvC/RgBYK+XOiP1YTknbzSCOneSRBzZrM2w4DUUdD3yI
```

```
sxx8Wy2O9vPJI8BD8KVbGI2Ou1WMuF04OzT9fBdXQ6MdGGzeMyEstSr/POGxKUAY
EY18hKcKctaGxAMZyAcpesqVDNmWn6vQClCbAkbTCD1mpF1Bn5x8vYlLIhkmuqui
XsNV6z3WFwACAg//fOYTkdIaae2KgiBhuMrXUo+rhdHy7Rq++gH7bA7KSMKtBsLg
+cRkv4yfV2NdoQArTFdA2dY14BKPPpsF32THU+IVEBzyJG4Ii9jUtmTDElWMzbKw
pt5lC1/FfDmjDmQdLAgcbmtviZ9B5deivfBTwD9zXgu2y8pYONVa89wkdc3lnUTF
81xBH9t5LJONJe7+bKECwbdXsulWOsHOLR6P31mSERAIE430jtz2/3WNOUX7cFqy
/39DdmTr6AesZBwZIUi8BFU533B/ZzS/YWh5FENK5WLuV32Ia4E3IjM3/8O6khI4
HnEnTIjON74aO5ElUbjQzP7RFrIYlxAdG1gU+uVBcE/7t6bObIa7O7sgeBQnzuyK
9aEWOaKwKd9uUN5Vqks3zKCu6MWvShyOEF3bJQKQ48268AGN7hcOlKuLkoRhkxYO
3PI9Wrn+ExW/58pVOpcb5opPo2tSlZIAK3ExXcEq/2DDL6KD2BLfseZa/l/HeZww
VBcKFyOqsIUkYklX7wyHIjNyunwnK6X+lUytwccjUOqHK+hTFCDUn7ZlYpaDhqt6
GxoKY4/AlWIr3K7OozldVF8Dua9gYg1ZMoYS7ElPwcNHjobcwI+IwVlSFDWLZHAq
ncaHkPUSm+EcpIMLZvuamZ9Yl1mlif7PeewZVqqQrkbPgA61+glp5a5wBQuJAEYE
GBECAAYFAjhaiyIACgkQ+e69xICJydpImACdFgr4Lba2LO+htIRrfXanZqKG1h4A
nOiMXY/mlu2Wp8Orne53HLvtMhei
=Nb3Y
-----END PGP PUBLIC KEY BLOCK-----

-----BEGIN PGP SIGNATURE-----
Version: PGP for Personal Privacy 5.5.2

iQA/AwUBOFqLufnuvcSAicnaEQJxwwCgsPWRtOmdIyQlIEGg4YrODM18qucAoNeb
pzrhsH4OxTLMfaXGnHGiO9Kj
=BEt4
-----END PGP SIGNATURE-----
```

Observați că este semnat, deci verific dacă într-adevăr este cheia mea. Vă rog să nu încercați să o introduceți de la tastatură. Dacă doriți să o testați, este disponibilă pe situl meu Web, la `http://www.leeanne.com/`.

Adesea se utilizează combinații ale ambelor metode, deoarece există numeroși algoritmi simetrici de criptare care sunt foarte rapizi, în timp ce algoritmii asimetrici sunt relativ lenți. În esență, metoda asimetrică cu cheie publică este folosită pentru a transmite cheia de criptare simetrică, iar apoi sesiunea continuă folosind metoda simetrică, mai rapidă.

Fiți conștient de faptul că majoritatea sistemelor de criptare aprobate de guvern, cum este DES, au fost proiectate intenționat astfel încât să fie ușor de spart. Cei mai buni algoritmi de criptare, cum este cel utilizat de PGP sau de puternicele metode RSA și Diffie-Hellman, sunt clasificați ca „muniție" de Guvernul SUA. Exportul acestor algoritmi, fără a intra în conflict cu reglementările proiectate evident pentru a preveni utilizarea lor casnică și a încetini răspândirea cunoștințelor de criptare, era riscant până când Departamentul de Comerț al SUA a permis în sfârșit exportul de produse comerciale de criptare, în ianuarie 2000. Deși aceste constrângeri la publicarea codurilor sursă de criptare au fost considerate recent ca fiind neconstituționale de către Ninth Circuit Court of Appeals, anulând întreaga lege, guvernul a intentat recurs, care va ajunge fără îndoială la Curtea Supremă, încercând astfel să păstreze dreptul de a controla aceste exporturi, chiar dacă sunt de acord cu faptul că secretul a fost deja divulgat.

Sugestie de la LeeAnne.com Words to weave by

Algoritmul DES pe 56 de biți a fost spart în două zile de un microcalculator creat special în acest scop. Deci puteți presupune că toate metodele care pot fi exportate legal nu vor ridica nici o problemă pentru supercalculatoarele Agenției Naționale de Securitate sau ale altei agenții guvernamentale.

Din acest punct de vedere, guvernul SUA este fie îngust la minte, fie, mult mai probabil, viclean. Mâța criptării a ieșit deja din sac. Ei nu o mai pot vârî la loc, indiferent cât de mult ar încerca, deoarece oricine este interesat de aspecte secrete sau de confidențialitate utilizează deja produse foarte sigure, multe dintre ele gratuite sau foarte ieftine. PGP-ul asimetric este un exemplu. O companie franceză are un cifru simetric, numit de ei CS-Cipher, care permite criptări până la 128 de biți, pe care îl vor vinde oricui. Ei au oferit un premiu de 10.000 de euro celui care îl sparge, deci poate doriți să aruncați o privire, dacă vă place să rezolvați puzzle-uri dificile. Există o versiune gratuită cu o securitate pe 40 de biți, dar pentru nivelurile superioare, ei percep o taxă.

De asemenea, ar mai trebui să rețineți că restricțiile de export nu fac de fapt nimic în practică în afară de hărțuirea producătorilor și utilizatorilor de produse criptografice. De exemplu, Netscape are versiuni de export și interne, cea internă având o criptare puternică, fiind disponibilă numai prin completarea unui formular destul de lung.

Restaurarea criptărilor puternice

Oricum, există situri în întreaga lume care restaurează criptări puternice ale versiunii de export incomplete a Netscape. Unul dintre ele este Fortify, la `http://www.fortify.net/`, iar altul este Safe Passage de la C2net, la `http://www.c2.net/`. De asemenea, C2net mai furnizează o versiune completă a SSL, Stronghold, deci produse puternice de criptare sunt disponibile în toată lumea, în orice situație. Criptografia avansează în mod constant. Ultimele dezvoltări conțin tehnici de criptare cuantice care fac interceptarea și decriptarea mesajului codificat imposibile din punct de vedere teoretic fără descoperirea unei noi realități fizice care să permită rezolvarea nesiguranței cuantice fără a afecta starea sistemului.

Criptați singuri fișiere

Sunt disponibile multe instrumente care criptează și decriptează on-line fișierele dumneavoastră, deci secretele companiei pot fi păstrate într-o manieră sigură, chiar dacă o intruziune fizică sau logică expune inamicilor pentru investigare conținutul hard-discului dumneavoastră. Unele sunt extrem de ușor de decodat, deși pot depăși capacitatea unui copil obișnuit de 12 ani, în cazul în care vreți să ascundeți ceva de copiii dumneavoastră. Unele sunt extrem de dificile, iar descifrarea lor ar asigura faima criptografului care le-a spart. Sunt disponibile atât de multe produse de criptare, încât ar fi necinstit să creăm impresia că RSA și PGP sunt singurele oferte valabile. O căutare folosind cuvintele cheie „cryptography" (criptografie) sau „computer security" (securitatea calculatoarelor) va găsi sute de produse.

Sugestie de la LeeAnne.com Words to weave by

Nu toate produsele de criptare au aceleași performanțe. Multe dintre ele oferă doar iluzia securității. Studiați alternativele înainte de a alege.

Criptare TCP/IP

Criptarea este și ea o parte a generației următoare a TCP/IP, cel puțin la nivelul IP. Implementarea Sun a IPv6, numită uneori IPng (IP next generation – IP generația următoare) conține un protocol numit SKIP, Simple Key-management for IP (gestionare simplă de chei pentru IP). SKIP utilizează un sistem de cheie publică Diffie-Hellman foarte inteligent pentru a cripta și a transmite scurte chei de pachete, care la rândul lor criptează datele conținute în pachete fără a adăuga prea multe cheltuieli suplimentare pentru fiecare pachet. Este un mecanism sigur, ale cărui valori se modifică frecvent pentru a elimina posibilitatea colectării de date suficiente pentru a începe să observați repetițiile. Acest nivel de criptare poate să se combine cu criptarea oferită de SSL, deci combinația celor două poate fi mult mai mare decât suma părților. Deoarece IP este fără conexiune, tot ce este diferit de o cheie foarte scurtă este o cheltuială mult prea mare, deoarece cheia trebuie să apară în fiecare pachet. Deci SKIP modifică cheia pachetului deseori, pentru a evita să ofere criptografilor prea mult text cu care să lucreze.

Utilizarea Internetului pentru a furniza comunicări private în acest fel este numită uneori *tunneling* și este baza pentru Virtual Private Networks (rețea virtuală privată), care utilizează Internetul pentru a lega sigur porțiuni ale unei rețele fără a dezvălui conținutul de text simplu al traficului efectiv prin rețea. Dacă nu sunt proiectate cu atenție, rețelele virtuale private sunt sensibile la o metodă de atac numită *analiza traficului* (*traffic analysis*), care presupune urmărirea numărului de mesaje și a cantității de date care sunt transmise pe o cale sigură. Prin observarea sau manipularea evenimentelor din exterior și apoi observarea mărimii fluxurilor de date dintre siturile legate, un criptograf poate face deducții bune cu privire la conținutul mesajelor, ceea ce face decriptarea oarecum mai ușoară.

Așa cum o limbă poate fi învățată prin observarea schimbului de cuvinte și de acțiuni repetate, un criptograf poate descifra codurile utilizate prin observarea discuției dintre nodurile rețelei virtuale.

Așa că sistemele foarte sigure mențin deseori un nivel de trafic constant, transmițând fără întrerupere mesaje aleatoare atunci când nu au de trimis mesaje efective. În ciuda faptului că acest lucru consumă din lățimea de bandă, are avantajul că nu poate fi observat nici un model de activitate, făcând astfel atacurile criptografice mult mai dificile. Deși criptograful rival poate încă să observe sau să manipuleze evenimentele din exterior pentru a ghici conținutul traficului, el nu are cum să știe care dintre multele mesaje conține de fapt acel trafic.

Inițiative de securitate și confidențialitate

Cea mai vestită inițiativă de securitate în contextul acestei cărți este, evident, P3P, studiat în capitolul anterior, dar ea nu a apărut din neant. *Marcajele de încredere* (*trust mark*), *software-ul de blocare controlat de utilizator* (*user-controlled blocking software*) și *serviciile de autentificare* (*authentication service*), fiecare referindu-se la diferite părți ale problemei, existau cu ani înainte, pe măsură ce furnizorii încercau prin intermediul grupurilor din domeniu să împiedice intervenția guvernului. P3P este numai unul din aceste grupuri. Furnizorul de servicii de confidențialitate electronică eTrust cheltuiește milioane de dolari pentru a-și face cunoscut programul de confidențialitate pe Internet, în timp ce Netscape are o pasiune pentru standardul de profiluri deschise (OPS – Open Profiling Standard), care permite utilizatorilor să trimită e-mail criptat și să își creeze „pașapoarte digitale" stocate pe propriile lor hard-discuri, care pot fi utilizate pentru a accesa automat siturile Web pentru cumpărături.

Toate firmele importante de contabilitate încep să își includă în program și comerțul electronic. Price Waterhouse, Coopers & Lybrand Consulting, Ernst & Young, Arthur Andersen și altele fie stabilesc relații cu furnizori ca Cyber Cash, preluând pentru clienții lor toate departamentele de comerț on-line, fie se gândesc, cel puțin, să o facă.

Comitetul de coordonare pentru Intercontinent Research Networking, un grup internațional, lucrează la o colaborare între grupuri de cercetare din diferite țări pentru a promova infrastructuri și politici de confidențialitate și securitate comune.

Rezolvarea problemei e-mail-urilor comerciale supărătoare (numite uneori *spam*, cu scuzele de rigoare față de Hormel) este în atenția mai multor organizații, printre care CAUCE, `http://www.cauce.org/`, și MAPS, `http://mail-abuse.org/`.

Chiar și Fiscul (Internal Revenue Service) s-a alăturat celor mai tari, cu propria caracteristică „Strategies for Growth" (strategii de dezvoltare), care susține cu putere preocupările pentru securitate și confidențialitate ale contribuabililor din SUA și furnizează metode care le asigură, concomitent cu dorința lor de a se ajunge repede la completarea electronică a declarațiilor de venit.

CERT (Computer Emergency Response Team), la `http://www.cert.org/`, publică sfaturi și primește rapoarte referitoare la încercările de atac. Este o resursă neprețuită pentru administratorii responsabili cu securitatea rețelelor și calculatoarelor.

Electronic Frontier Foundation, `http://www.eff.org/`, înregistrează atacurile asupra propriei noastre confidențialități efectuate în primul rând de către organizațiile guvernamentale, deși se pot urmări și invaziile comerciale și ale altor entități. Electronic Privacy Information Center, la `http://www.epic.org/`, cercetează mai larg și mai atent intrușii guvernamentali și particulari care ne pun în pericol confidențialitatea.

Cryptography Research are o listă extinsă de legături la resurse criptografice de pe Web, la `http://www.cryptography.com/`.

Cookie-urile și alte formulare care conțin date personale și de comportament pot fi învinse manual sau prin software disponibil pe multe platforme. Exemplele includ Internet Junkbuster Proxy, disponibil gratuit la `http://www.junkbusters.com/ht/en/ijb.html`, pentru Windows 95/98/NT și UNIX, care blochează atât cookie-urile, cât și bannerele publicitare sub Windows. Altul este interMute, la `http://www.intermute.com/`, care blochează cookie-urile, bannerele publicitare și ferestrele pop-up, dar rulează numai sub Windows. O soluție bazată pe Java, numită Muffin, la `http://muffin.doit.org/`, este disponibilă, dar este destul de dificil de configurat. Totuși este gratuită, deci este greu (sau cel puțin ingrat) să te plângi.

Mulți oameni nu știu că furnizorii de bannere publicitare pot să urmărească mișcările dumneavoastră pe un sit Web, sau chiar de la un sit la altul, prin plasarea unui cookie pe calculatorul dumneavoastră. De fiecare dată când ajungeți la un sit servit de același furnizor de bannere, se poate afla cine sunteți dacă cookie-ul este disponibil pentru ei. Deoarece „extragerea datelor" (alt cuvânt pentru monitorizarea clicurilor) este foarte valoroasă pentru advertiseri, cunoașterea locațiilor accesate prin navigarea de la un sit la altul este o fereastră cu vedere la interesele și chiar la concepțiile dumneavoastră.

Figura 16.5 prezintă ecranul jurnal interMute afișat peste o pagină de căutare Altavista. Observați că pe pagina Altavista nu există nici un banner, iar jurnalul prezintă 24 de ban-

nere care ar fi fost afișate dacă nu s-ar fi utilizat acest instrument. Într-un mediu de lucru, economia de timp care ar fi fost necesar pentru descărcarea bannerelor, fără a mai menționa întreruperea în cazul în care un angajat este ademenit de un banner și execută un clic pe el, ar putea să amortizeze în câteva luni prețul acestui produs.

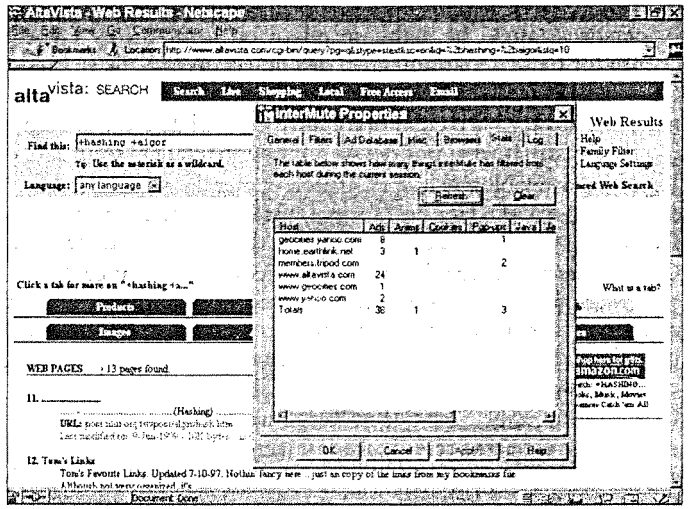


FIGURA 16.5
Ecranul jurnalului interMute afișează numărul de bannere blocate într-o sesiune foarte scurtă.

Semnături digitale

O semnătură digitală este o metodă de atașare a unui identificator unic, care nu poate fi negat ulterior. Asemenea semnăturilor scrise de mână, o semnătură digitală ar trebui într-o oarecare măsură să se auto-identifice.

Istoria semnăturilor digitale este formată din mai multe părți. În primul rând, funcțiile hash oferă o modalitate de a crea un număr *aproape* unic, care elimină aproape toate informațiile conținute în document care nu aparțin de identitatea sa. Așa cum o amprentă vă spune despre o persoană doar faptul că este unică între celelalte persoane, o funcție hash condensează informația și structura unui document într-un număr relativ mic, care poate identifica documentul între multe altele, cu o probabilitate extrem de mare.

Cu cât este mai bună funcția hash, cu atât este mai probabil ca numărul rezultat să fie într-adevăr unic.

Deci prin aplicarea unei funcții hash la un mesaj și prin includerea funcției într-un mesaj criptat utilizând o cheie privată, oricine are cheia publică poate decripta semnătura codificată și poate obține condensarea hash a mesajului original. După încă o aplicare a funcției hash, rezultatele pot fi comparate, iar mesajul în sine poate fi verificat dacă este identic cu mesajul care a fost semnat.

XML semnat

Grupul de lucru W3C pentru semnăturile digitale XML, la momentul scrierii acestei cărți, este încă într-o fază preliminară a definirii unui standard, dar semnăturile digitale sunt înțelese destul de bine, astfel încât este probabil ca recomandarea finală să semene foarte mult cu schița.

Deoarece într-un context XML este posibil să fie nevoie ca multe obiecte digitale să fie semnate, schița de lucru nu impune restricții cu privire la conținutul documentelor semnate. XML Signature este proiectat să asigure integritatea oricărui tip de document, chiar dacă nu este XML, indiferent dacă se află la distanță sau nu. De asemenea, el servește la autentificarea mesajelor și a semnăturilor asociate unui document.

Deoarece documentele XML pot fi modificate drastic prin foi de stil, filtre, includerea de imagini și prin includerea ca notații a altor conținuturi care nu sunt XML, XML Signature capturează întregul context al documentului așa cum este acesta în momentul semnării. Părțile invizibile ale documentului nu sunt semnate și, implicit, integritatea lor nu este garantată.

Deși sintaxa exactă nu este încă fixată, o semnătură XML va semăna probabil cu aceasta, preluată din documentul de sintaxă al ultimei schițe de lucru de la W3C:

```
<Signature>
  <SignedInfo>
    (CanonicalizationMethod)?
    (SignatureMethod)
    (<Reference (URI=|IDREF=)? Type=?>
      (Transforms)?
      (DigestMethod)
      (DigestValue)
    </Reference>)+
  </SignedInfo>
  (SignatureValue)
  (KeyInfo)?
  (Object)*
</Signature>
```

unde metoda opțională `CanonicalizationMethod` descrie metoda utilizată pentru a obține forma canonică, adică extinderea tuturor referințelor de entități la forma completă și realizarea altor transformări dorite pentru a asigura că documentele care par să fie identice sunt de fapt echivalente. `SignatureMethod` descrie, de exemplu, tipul exact al semnăturii care va fi generată, iar articolele `Reference` indică părțile efective ale documentului care sunt semnate. Celelalte articole descriu semnătura efectivă și datele legate de aceasta.

În ciuda faptului că acest lucru poate părea complex, conceptul de bază este simplu; construiți o valoare hash care descrie datele, criptați-o într-un mod reversibil utilizând o cheie privată asociată unui algoritm cu cheie publică și împachetați totul. Oricine va dori să verifice documentul sursă poate recalcula valoarea hash și apoi să decripteze valoarea originală semnată și criptată utilizând cheia publică potrivită cheii private. Dacă valorile sunt identice, probabil că documentele sunt identice.

Limbaje pentru descrierea marcajului semnat (SMDL – Signed Markup Description Language)

Semnăturile digitale, așa cum au fost definite anterior în acest capitol, sunt acum în centrul atenției, iar în curând vor fi aplicate la tot ce mișcă pe Web. Există deja inițiative de a adăuga semnături digitale la HTML, XHTML și chiar la grafica pe calculator și la alte fișiere multimedia. Acest lucru este făcut pe de o parte pentru a asigura stabilitatea conținutului și pe de altă parte pentru a rezolva preocupările referitoare la proprietatea

intelectuală în cazul conținutului multimedia plasat pe Web. Deși tehnologii ca RealAudio au avut câteva încercări de a ascunde locația efectivă a conținutului, oricine poate lua aceste fișiere de la locația lor și să le utilizeze din nou, în mod ilegal, pentru alte scopuri. Totuși, deteriorarea cu un sunet a unei melodii ar duce la compromiterea melodiei, la fel și mâzgăliturile de identificare pe o operă de artă sau pe o fotografie.

Steganografia (*steganography*) este o metodă de ascundere a datelor criptate într-un mod care nu poate fi detectat, chiar dacă bănuiți că au loc comunicații secrete. Ea utilizează spațiul informațiilor nefolosite, existente în aproape orice fișier, și poate fi făcută suficient de redundantă astfel încât informația secretă să supraviețuiască la copiere, decupare, editare sau la altă modificare, împreună cu fișierul gazdă care o înfășoară.

Deci steganografia, știința ascunderii de informații criptate în mici variații de lumini sau umbre, niveluri de sunet sau în orice altă variabilă cu proprietăți statistice care poate fi manipulată, oferă muzicienilor, fotografilor sau altor persoane a căror muncă poate fi furată cu ușurință pe Web și care, odată furată, poate fi găsită cu greu, o metodă de a dovedi dreptul legal de proprietate. Propoziția precedentă prezintă o formă foarte primitivă de steganografie, prin codificarea cuvântului „ascuns" în literele **aldine** dintr-un text simplu. Într-un mesaj steganografic real, variațiile din text ar fi atât de mici încât ar putea fi detectate numai dacă ați ști ce să căutați, iar mesajul în sine ar putea și el să fie criptat.

Atunci când informațiile steganografice sunt utilizate pentru a furniza informații de identificare și de drepturi de autor, ele se numesc de obicei filigran digital, sau pe scurt filigran, și există mulți furnizori de produse steganografice.

Una dintre cele mai complete liste de legături și informații steganografice se află în paginile de informații și arhive despre steganografie ale Dr. Eric Milbrandt (Dr. Eric Milbrandt's Steganography Info and Archive pages), la `http://members.tripod.com/steganography/stego.html`, unde acesta întreține o colecție imensă de informații, legături la software și multe altele. Multe pachete steganografice sunt fie shareware ieftin, fie sunt gratuite.

Metode mult mai sofisticate sunt posibile, iar instrumente steganografice au fost încorporate în multe medii profesionale de creare și editare de grafică și audio. De exemplu, Adobe Photoshop vă permite să adăugați filigrane digitale puternice la lucrările dumneavoastră, alese dintr-un meniu, utilizând produsul steganografic Digimarc. Figura 16.6 prezintă o vizualizare exagerată și foarte mult mărită a ceea ce se întâmplă atunci când adăugați un filigran. Imaginea de bază era un ecran alb simplu, deci modificarea va fi vizibilă, dar ajustările fiecărui pixel au fost atât de mici încât nu puteau fi văzute fără modificarea imaginii în sine pentru a mări variațiile foarte slabe în strălucire la o imagine alb-negru cu un contrast de 100% între albul pur și orice altceva.

Figura 16.7 prezintă pagina introductivă a unui furnizor de produse steganografice ale cărui pretenții entuziaste au depășit realitatea, deoarece este evident că imaginea poate fi și chiar a fost copiată. Presupun că în imagine există un filigran digital și că acesta a fost scanat din pagină și identificat – cel puțin așa ar trebui să fie. Pretențiile referitoare la imposibilitatea copierii se bazează în întregime pe mediul browserului și pe concepțiile cuiva despre copiere. De fapt, imaginea din dreapta nu se afișează deloc, decât dacă sunt activate script-urile, lucru evitat de multe persoane prevăzătoare, iar așa-numita „protecție la copiere" se bazează pe încercarea de a copia imaginea într-un mod stereotip. Dacă cineva gândește în afara parametrilor „normali", copierea imaginii este o sarcină banală.

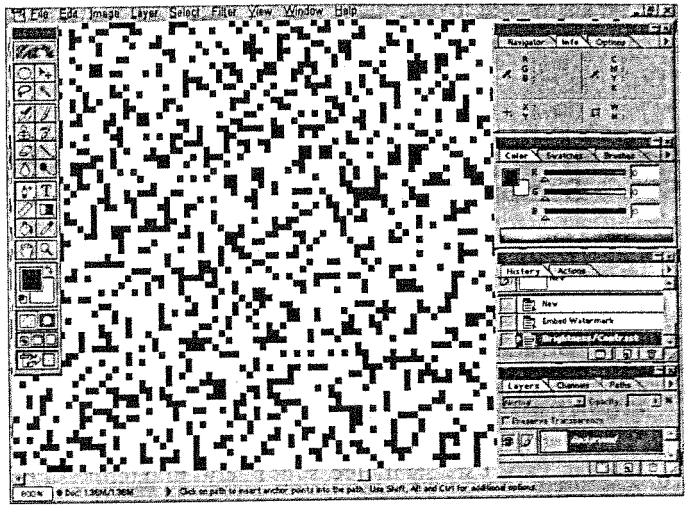


FIGURA 16.6
Un fragment înglobat de cod steganografic efectiv, vizualizat cu Adobe Photoshop pentru a permite ca modificările să fie vizibile.

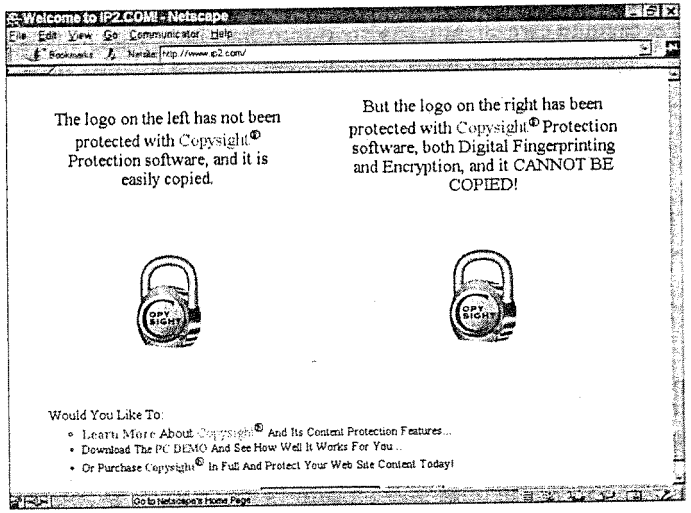


FIGURA 16.7
Pagina introductivă a unui vânzător de produse steganografice prezintă rezultatul obținut dacă vorbești fără să gândești.

Insist puțin pe această imagine deoarece vă prezintă instrumentul cel mai important, atât din arsenalul crackerului, cât și din trusa de prim-ajutor a administratorului responsabil cu securitatea. Imaginația. Luând-o pe căi lăturalnice, crackerii fac lucruri pe care ar trebui să nu le poată face. Un administrator de securitate vigilent preîntâmpină atacurile crackerilor și hackerilor sau rezolvă problema rapid dacă are loc un atac, imaginând metode care nu sunt standard.

De fapt, jocul cu script-uri Java complicate pentru a afișa o imagine și pentru a pretinde că împiedici copierea ei este o pierdere enormă de timp și de lățime de bandă, deoarece este o problemă banală să dobori astfel de scheme. Dacă poți să *vezi* imaginea, o poți captura din buffer-ul de afișare al calculatorului, apăsând o singură tastă. Oare de ce se

complică unii cu astfel de prostii dacă, în același timp, imaginea este inaccesibilă pentru persoanele care nu folosesc browserele în modul în care se așteaptă? Acest lucru mă depășește. Cred că din aroganță, orgoliu și nechibzuință, în aceeași măsură.

▶ Pentru mai multe informații despre Java și accesibilitate, **vezi** „Folosirea JavaScript ca motor de redare", la **pagina 592.**

SSL – Secure Sockets Layer

Criptarea este o parte importantă a securității oferite de SSL (Secure Sockets Layer) de la Netscape, care este suportat de majoritatea furnizorilor importanți de servere, printre care, evident, Netscape, Microsoft și Apache. TLS (Transport Layer Security), care este un succesor al SSL, este un standard oficial al IETF.

Procesul SSL arată astfel:

1. Clientul trimite o cerere de conectare la serverul sigur, asemănătoare cu aceasta: `https://numeserver.domeniu.com`.
2. Serverul trimite clientului un certificat autentificat prin pre-semnare și care nu poate fi negat. Pașii 1 și 2, la un loc, sunt referiți prin termenul *handshake* (*dialog de confirmare*).
3. Clientul verifică dacă certificatul a fost emis de o autoritate în care are încredere. În cazul în care clientul găsește o autoritate în care are încredere, el trece la pasul următor. În caz contrar, fie anulează conexiunea, fie continuă în nesiguranță, fără autentificare. Sau, clientul poate întreba utilizatorul dacă se poate avea încredere în autoritatea care a propus certificatul, iar sesiunea continuă în consecință.
4. Clientul compară informația din certificat cu informația primită de la sit – numele domeniului său și cheia sa publică. Dacă informația se potrivește, situl este autentificat.
5. Clientul îi spune serverului ce scheme de criptare și tipuri de chei de criptare poate utiliza.
6. Serverul alege cea mai puternică schemă de criptare pe care o partajează cu clientul și îi spune acestuia care schemă va fi utilizată.
7. Utilizând schema de criptare aleasă, clientul generează o cheie a sesiunii (o cheie de criptare simetrică, utilizată numai pentru această tranzacție) și o criptează utilizând cheia publică a serverului.
8. Clientul trimite serverului cheia criptată a sesiunii.
9. Serverul recepționează cheia criptată a sesiunii și o decriptează folosind cheia sa privată.
10. Clientul și serverul utilizează aceeași cheie de sesiune pentru a cripta și a decripta datele pe care le trimit și le recepționează.

Deoarece date necriptate sunt trimise doar la începutul sesiunii, pe durata dialogului de confirmare și a negocierii criptării, metoda este foarte sigură dacă schema de criptare în sine este sigură. Din păcate, DES (criptare pe 56 de biți) sau unul mai slab este cel mai mic numitor comun al multor asemenea sisteme, deci nivelul efectiv de securitate poate varia dinamic de la server la server și de la foarte bun la foarte rău.

Notă Orice schemă de securitate care se bazează pe certificate, ca și aceasta, ar trebui să țină cont de faptul că certificatele sunt proiectate să expire la un moment dat. În ultimă vreme, milioane de certificate au expirat (conform proiectării) cu trecerea Y2K, deci utilizatorii au fost nevoiți să își actualizeze browserele și serverele cu certificate noi pentru a realiza în continuare tranzacții sigure.

Totuși, nici acest lucru nu este perfect, chiar și cu criptări puternice. Există o tehnică numită *hyperlink spoofing* care poate fi utilizată pentru a ataca tranzacția, făcând clientul să creadă că el comunică direct cu un server sigur, când de fapt a fost dus de nas și comunică printr-un proxy care efectuează un spoofing (spionare). Codul hash MD5 utilizat la autentificarea mesajelor are și câteva breșe de securitate, deci este posibil să genereze mesaje de înlocuire care ajung la aceeași valoare după aplicarea funcției hash.

Pe de altă parte, spargerea SSL, chiar și la nivelul DES pe 56 de biți, este o operație extraordinară. Nu este genul de lucru pe care îl faceți dintr-o toană în timpul pauzei de masă. Ar fi bine să aveți la cheremul dumneavoastră un supercalculator asemenea celor de la Agenția Națională de Securitate. Iar atunci când sunt utilizați puternicii algoritmi RSA, care permit un nivel de criptare mult mai înalt, este o sarcină imposibilă în momentul de față. Lungimea de biți a schemelor de criptare este o prescurtare a descrierii complexității transformării prin descrierea lungimii cheii utilizate pentru a crea textul cifrat. Din punct de vedere criptografic, aceasta nu este o măsură adevărată a siguranței, dar este suficientă pentru a măsura aproximativ, presupunând că criptanaliștii și-au făcut treaba.

Sugestie de la LeeAnne.com Words to weave by

Lungimea de cheie minimă necesară pentru siguranță este considerată de obicei ca fiind în jurul a 128 de biți, dar există cinici care cred că sunt necesare coduri pe 256 de biți sau mai mult pentru a asigura că marele secret din acest an nu va deveni anul viitor o știre veche. Știința înaintează, nu-i așa?

O dată cu creșterea puterii calculatoarelor, lungimea minimă a cheii necesare pentru o securitate deplină crește fără limite.

Funcții hash

Funcțiile hash sunt metode prin care se verifică validitatea sau chiar identitatea unui document sau a unei operații matematice. Cea mai familiară este operația de „distribuire a cifrei 9" pe care o foloseați la școală pentru a verifica dacă ați adunat corect o coloană de numere. Această funcție hash se bazează pe clasele de resturi modulo-9 și irosește destul de multe informații despre numere, dar identifică majoritatea erorilor de calcul.

O dată cu îmbunătățirea funcțiilor hash, cantitatea de date pe care o rețin a crescut, dar toate funcționează prin risipirea datelor într-un mod algoritmic, reținând un fel de „amprentă digitală" a datelor originale care poate fi utilă în mai multe moduri.

Un anumit tip de funcție hash este folosit pentru a verifica dacă un flux de date a fost alterat într-un anumit mod. Când este utilizată pentru a verifica validitatea fluxului de date, valoarea hash este numită uneori CRC, cyclic redundancy check (verificare ciclică prin redundanță). CRC-urile funcționează prin calcularea unei valori pe loc, în funcție de valoarea fiecărui caracter transmis. La sfârșitul transmisiei, se transmite și valoarea calculată. Procesul receptor folosește aceeași funcție pe măsură ce datele sosesc și, prin compararea valorii calculate de expeditor cu valoarea proprie, poate identifica orice eroare simplă din transmisie.

Aceste tipuri primitive de funcții hash pot fi ocolite dacă există mai multe erori care se anulează reciproc. Deci, CRC-urile se calculează uneori în mai multe moduri, fiecare fiind vulnerabil sau puternic în anumite privințe, astfel încât puține greșeli scapă nedepistate.

Verificarea unui document pentru a vă asigura că nu a fost modificat este o variantă simplă a schemei CRC, dar cu un număr mai mare de metode hash mai complexe, care asigură faptul că orice schimbare în document este reflectată într-o valoare hash diferită. Deci în timp ce un CRC simplu poate avea o lungime de unu sau doi octeți, permițând ca numeroase transmisii să se reducă la aceeași valoare, o valoare hash care identifică un document ca neschimbat trebuie să fie mult mai mare, astfel încât orice document din lume să dea o valoare hash diferită.

De exemplu, MD5 generează un mesaj rezumat de 32 de cifre hexazecimale, adică 128 de biți în lungime, aproximativ 3×10^{38}. Este un număr atât de mare încât nici Scrooge McDuck nu a avut atâția bani. Sunt posibile mai multe documente decât firele de nisip de pe toate plajele din lume.

Succesorul lui MD5, Secure Hash Algorithm (SHA), generează un număr de 160 de biți, deci este chiar mai sigur decât MD5.

P3P

P3P este doar o mică parte din întregul puzzle al securității și a fost tratat în detaliu în Capitolul 14, dar merită să repetăm: datele dumneavoastră personale sunt o parte importantă a securității dumneavoastră. Majoritatea oamenilor nu au secrete nucleare pe calculatoarele proprii. Informațiile personale sunt singurele lucruri pe care le au și care merită să fie furate. Dar într-adevăr *merită*. De fapt, probabil sunt furate de câte ori sunteți on-line.

Depanarea problemelor de securitate

Un exemplu de securitate pe calculator

Să presupunem că un hacker a intrat pe calculatorul dumneavoastră. Fie a făcut unele pagube, fie a lăsat un mesaj sarcastic pe consolă, sau, inspectând jurnalele, v-ați dat seama că a apărut o logare neautorizată ca root sau administrator.

Mai întâi întrerupeți legătura Internet. Faceți acest lucru imediat, prin metoda cea mai drastică, cum ar fi deconectarea cablului Ethernet din spatele calculatorului. Este foarte posibil ca hackerul să fie încă pe calculator, sau să se reconecteze mai târziu, sau să producă alte pagube în timp ce încercați să diagnosticați și să remediați sistemul dumneavoastră. Deci eliminați criza și calmați situația.

Este momentul să căutați urmele. Acest lucru poate fi surprinzător de dificil. Primele elemente înlocuite pe sistemele care au fost sparte cu succes sunt chiar instrumentele de sistem de care veți avea nevoie pentru a depista problemele potențiale. Dacă sunt mai multe calculatoare în rețea, merită să urmăriți și să înregistrați traficul rețelei de pe toate sistemele. Analiza traficului de rețea la momentul sau în jurul momentului în care a avut loc spargerea poate ajuta la depistarea sursei efective a atacurilor.

Pe sistemele UNIX, comenzile `ls` și `ps`, ca și multe alte instrumente de sistem, sunt adesea înlocuite cu versiuni care încorporează un blocaj pentru fișierele și procesele hackerului. Pe sistemele Windows, comanda `dir` și alte utilitare Windows pot fi modificate într-un mod asemănător. Un program bun de detectare a virușilor poate adesea să depisteze fișierele care au fost modificate, dacă ați avut inspirația să realizați o semnătură hash pentru fiecare fișier al sistemului încă intact, dar trebuie să păstrați fișierul semnătură într-un loc sigur, off-line, altfel și acesta poate fi compromis.

Deci primul dumneavoastră pas ar fi să scoateți trusa anti-hacker, care conține copii sigure ale fișierelor și utilitarelor importante. Pentru UNIX, ar trebui să păstrați măcar `find`, `ps`, `ls`, `cp`, `rm`, `mv`, `gdb`, `nm`, `strings`, `file`, `strip` `(gnu)tar`, `zip`, `grep`, `less` sau `more`, editorul dumneavoastră favorit, programul shell favorit, programul antivirus, fișierele de date și `mount`. Desigur, aveți nevoie de copii sigure ale distribuției dumneavoastră și de copiile de siguranță ale fișierelor, deoarece va trebui să reparați stricăciunile și să restaurați fișierele.

Pe majoritatea sistemelor Windows veți dori în principal instrumente la nivelul DOS, deoarece sunt programe independente pe care le puteți folosi într-un mod controlat. Încercarea de a depista fișiere modificate într-o sesiune Windows atunci când nu puteți avea încredere în nici unul dintre numeroasele fișiere care alcătuiesc mediul de rulare Windows este un exercițiu de inutilitate. Vă vor fi necesare: un disc de încărcare, `dir`, `find`, `attrib`, `chkdsk`, un editor (să sperăm mai bun decât `edit`), discul de încărcare a programului antivirus, distribuția Windows pe CD-ROM și orice alte instrumente care vă pot ajuta. În cazul Windows, fie va trebui să restaurați totul de pe copiile de siguranță, fără a fi foarte sigur că acestea sunt într-adevăr curate, fie să reinstalați totul și să restaurați cu grijă din copiile de siguranță numai fișierele de date.

Instrumente de securitate din sistemul Windows

Windows este o țintă îndrăgită de hackeri, deoarece majoritatea calculatoarelor din lume rulează o versiune Windows în acest moment, iar slăbiciunile sale sunt foarte cunoscute. Deși versiunile mai vechi de Windows erau destul de nesigure și ușor de spart, prin dezvoltarea lui Windows NT și prin recenta lansare a versiunii Windows 2000, care se bazează parțial pe Windows NT, imaginea securității s-a îmbunătățit pentru utilizatorii Windows.

Gama largă de instrumente disponibile pentru administratorul de sistem UNIX nu este disponibilă nici sub DOS, nici sub Windows, decât în cazul în care ați achiziționat un pachet de instrumente add-on de la o companie cum ar MORTICE Kern Systems, la `http://www.mks.com` /. MORTICE Kern Systems dublează numeroase instrumente UNIX, și chiar o imitație de X-Windows care rulează în Windows NT, 98 și 2000. MKS Toolkit standard conține o mare parte din interfața standard UNIX, în timp ce MKS Toolkit Select adaugă instrumentele grafice, inclusiv un server X-Windows. Deși aceste instrumente nu vor fi familiare, și nici comode pentru administratorii care sunt obișnuiți numai cu un mediu Windows, puterea comenzilor UNIX îi va determina pe unii să își lărgească orizonturile.

În ambele cazuri, verificarea atentă a dimensiunilor fișierelor și a listingurilor tuturor directoarelor de pe calea de execuție poate să trădeze programe care nu ar trebui să existe. Căutați în special ciudățenii cum ar fi un fișier sau un director numit „...“ sau programe cu dimensiuni modificate. Dacă este posibil, jurnalele de consolă ar trebui să fie direcționate spre o imprimantă printr-o bifurcare, deoarece prima acțiune a unui hacker care obține un acces de root sau de administrator este să rescrie jurnalele de consolă și de sistem pentru a elimina cât mai mult din urmele atacului.

Este destul de obișnuit ca hackerii care au avut succes să lase o semnătură ascunsă undeva pe calculator, care să apară la depanare sau informații într-un executabil care să ofere indicii legate de identitatea hackerului. Este o idee bună să depistați și să salvați toate fișierele noi sau modificate pentru o inspecție ulterioară și o posibilă acțiune judiciară. Fișierele modificate și maniera de atac sunt foarte bine de știut, astfel încât să vă puteți proteja sistemul de alte atacuri similare. După ce sistemul a fost compromis, este foarte probabil să fie atacat din nou.

Înainte de a închide sistemul ar trebui să studiați cu atenție procesele care rulează. Se obișnuiește adesea să se instaleze un program ascuns, care încearcă să urmărească și să intercepteze încercările de logare pentru a colecta cât mai multe parole. Toate parolele trebuie să expire imediat. Nu aveți de unde să știți dacă o încercare de logare a fost interceptată și nu puteți să vă bazați pe utilizatori că își vor schimba din proprie inițiativă o parolă îndrăgită.

Ar fi chiar indicat să stabiliți parole noi și să le comunicați utilizatorilor printr-un canal sigur. De exemplu, scrisorile tipărite și recomandatele sunt destul de dificil de interceptat și sunt foarte potrivite pentru a comunica parole și chei. Orice logare la nivel de sistem sau rar utilizată trebuie să primească acest tratament, iar orice logare ocazională trebuie verificată cu grijă pentru a vă asigura că drepturile și grupurile de utilizatori nu au fost modificate. De fapt, toate logările ar trebui să fie verificate regulat pentru a depista promovări misterioase în autoritatea sistemului.

Utilizarea facilităților server-side – Java

CAPITOLUL 17

În acest capitol

De ce este atât de util Java pe server
Setarea serverului dumneavoastră Web și a software-ului aferent
Un program CGI simplu care emite cod XML
O abordare servlet
Transformarea pe server a codului XML în HTML
Un applet care cere cod XML
Utilizarea soclurilor în locul HTTP
CORBA
Depanarea Java și XML

De ce este atât de util Java pe server

Java este un limbaj simplu, considerat de mulți oameni mai ușor de învățat decât C++ sau Perl. Biblioteca de clase care însoțește toate compilatoarele Java include clase care manevrează comunicarea în rețea, manipularea imaginilor, accesul la baze de date, structuri multi-threading și multe altele. Acest lucru simplifică și accelerează mult codarea operațiilor server obișnuite.

Java s-a bucurat de multă atenție ca un limbaj pentru mai multe platforme. La prima vedere, este posibil să vă gândiți că un limbaj pentru mai multe platforme nu prea este un avantaj pentru programarea server-side. La urma urmelor, știți ce platformă utilizează serverul dumneavoastră, spre deosebire de browserele clienților care vizitează situl dumneavoastră Web. Dar alegerea unei soluții de server pe mai multe platforme înseamnă că nu va trebui niciodată să vă îngrijoreze mutarea sitului dumneavoastră pe o altă platformă. Deseori, siturile Web se mută de pe o platformă pe alta din diverse motive, cum ar fi scalabilitatea, costurile sau politica firmei. Este un lucru bun să știți că dacă o decizie externă vă obligă să schimbați servere, codul dumneavoastră Java nu trebuie recompilat.

Pentru mulți dezvoltatori, motivul pentru care utilizează Java pe server este că pot dezvolta servlet-uri. Acestea sunt înlocuiri comode pentru CGI și scalează mult mai bine decât CGI, mai ales Perl CGI. Servlet-urile vor fi discutate mai amănunțit în acest capitol.

Evident, Java este un limbaj orientat spre obiecte, iar XML și obiectele par să lucreze de minune împreună. Un obiect care se poate auto-construi dintr-un flux XML – sau care poate emite un flux XML care să îi descrie conținutul – se potrivește perfect la o soluție

server-side scrisă în Java. Un astfel de cod aproape că se scrie singur, după cum se va vedea și în cazurile studiate în acest capitol.

Și, în încheiere, unul dintre motivele care fac scrierea de programe Java pentru XML atât de atractivă este că puteți obține analizoare foarte bune – eficiente, suportând atât SAX cât și DOM, documentate acceptabil – în mod gratuit. În acest capitol veți lucra cu XML4J, un produs al proiectului AlphaWorks de la IBM, lansat ca open-source.

Setarea serverului dumneavoastră Web și a software-ului aferent

Dacă veți scrie programe Java care să ruleze pe server, primul lucru de care veți avea nevoie este un server. Punând totul la socoteală, trebuie să instalați și să configurați

- Un server Web – Personal Web Server sau IIS de la Microsoft, sau Java Web Server de la Sun.
- Java Development Kit (kitul de dezvoltare Java) – versiunea 1.1.8 sau mai nouă.
- O implementare de referință Java Server Web Development Kit, versiunea 1.01.
- Un analizor XML Java – XML4J de la IBM.
- Câteva drivere pentru baze de date, dacă aveți de gând să lucrați cu baze de date.

Software pentru serverul Web

Puteți utiliza orice software pentru servere Web care vă place sau se întâmplă să îl aveți instalat, dar acest capitol prezintă numai instrucțiunile de configurare a unora foarte populare. De la acestea, ar trebui să fiți capabil să extrapolați la serverul pe care îl utilizați dumneavoastră. De asemenea, ar trebui să aveți instalat JDK, sau un alt compilator Java, atât timp cât sunteți sigur că acesta produce cod pur JDK.

Mulți utilizatori de NT sau Windows 9x au acces la Internet Information Server (IIS) de la Microsoft. Dacă dumneavoastră nu aveți acces la acesta, atunci puteți descărca Microsoft Personal Web Server (PWS) de la `http://www.microsoft.com/Windows/ie/pws/default.htm` și să îl utilizați pentru exemplele din acest capitol. (Personal Web Server este parte a NT Option Pack, denumire prost aleasă deoarece funcționează și pe Windows 95 și Windows 98, nu numai pe Windows NT. Nu trebuie să descărcați tot Option Pack dacă doriți numai Personal Web Server.) Urmați instrucțiunile de instalare furnizate de Microsoft, cu o singură excepție. În mod prestabilit, programul de configurare al Personal Web Server nu instalează Internet Service Manager, pe care îl veți utiliza pentru ca programele Java CGI realizate de dumneavoastră să fie executate corect. Atât IIS cât și PWS sunt configurate în același mod, iar acest capitol prezintă modul în care le puteți configura pentru a utiliza pe server Java și XML.

Figura 17.1 prezintă ecranul în care selectați componentele din Option Pack pe care vreți să le instalați. Selectați Personal Web Server și executați clic pe Show Subcomponents. Va fi afișată caseta de dialog vizibilă în Figura 17.2. În această casetă, asigurați-vă că Internet Service Manager este selectat. Continuați instalarea. Dacă ați instalat deja Personal Web Server fără Internet Service Manager, atunci alegeți Start, Programs, Windows NT Option 4.0 Pack, Windows NT Option 4.0 Pack Setup și apoi instalați Internet Service Manager.

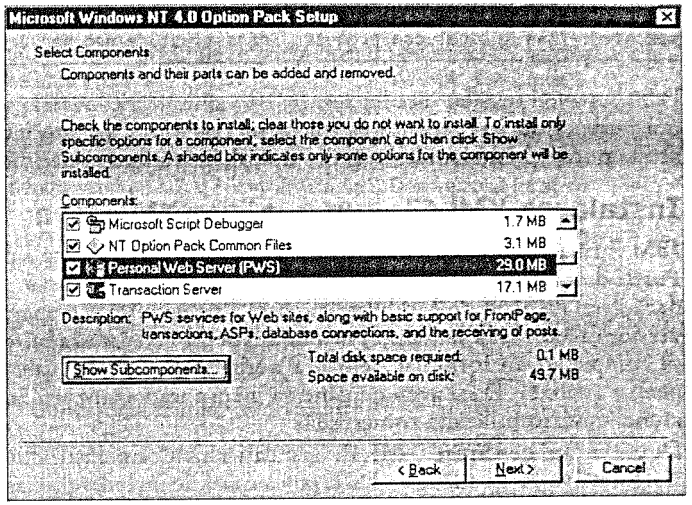


FIGURA 17.1
Când instalați NT Option Pack, aveți nevoie de o subcomponentă care nu este inclusă în instalarea prestabilită.

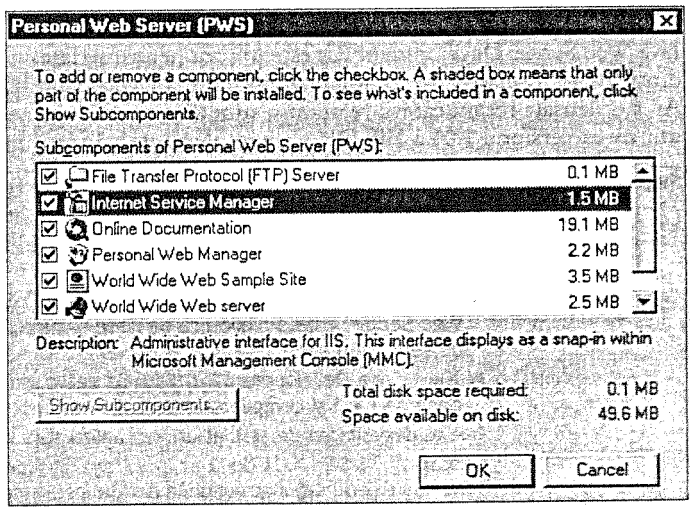


FIGURA 17.2
Asigurați-vă că Internet Service Manager este instalat atunci când instalați Personal Web Server.

Nici IIS și nici PWS nu suportă servlet-uri Java, deci poate veți prefera să utilizați în locul lor Java Web Server. Java Web Server rulează pe NT, Windows 9x, UNIX și pe orice altă platformă care suportă Java. O copie de încercare, cu durată limitată de utilizare, pe care să o utilizați pentru exemplele din acest capitol, o puteți descărca de la `http://www.sun.com/software/jwebserver/index.html`. Urmați instrucțiunile de instalare furnizate de Sun. În acest capitol se presupune că lucrați cu Java Web Server versiunea 2.0.

Java Web Server va fi înlocuit de iPlanet

Ca rezultat al alianței între Sun și AOL (care deține Netscape), Sun și Netscape nu vor mai face servere Web separate. În schimb, Java Web Server și Netscape Enterprise Server vor fi înlocuite de iPlanet Web Server Enterprise Edition.

iPlanet suportă servlet-uri şi este disponibilă o versiune de probă. Pentru mai multe informaţii, vezi `http://www.iplanet.com/products/infrastructure/web_servers/`.

Dacă nu aveţi instalat nici un server Web şi nu doriţi să testaţi folosind pagini HTML obişnuite, atunci puteţi utiliza micul server Web care este inclus în JWSDK. Despre acesta puteţi citi mai multe în acest capitol, în secţiunea referitoare la instalarea JWSDK.

Instalarea XML4J – un analizor XML Java

IBM furnizează analizorul său XML Java, XML4J, ca parte a proiectului AlphaWorks. Puteţi descărca gratuit analizorul de la `http://www.alphaworks.ibm.com/tech/xml4j`, chiar dacă aveţi de gând să îl utilizaţi ca parte a unei aplicaţii comerciale. Descărcaţi XML4J versiunea 3.0.1 (sau una puţin mai nouă) pentru a o utiliza în acest capitol. Vi se va cere să completaţi un formular pentru a obţine o licenţă de evaluare (Evaluation License) pentru analizor. Dacă aveţi de gând să încorporaţi analizorul într-un produs comercial, atunci descărcaţi licenţa comercială.

După ce aţi descărcat XML4J, instalaţi-l conform instrucţiunilor din readme.html. Asiguraţi-vă că aţi adăugat `xml4j.jar` la calea de includere a claselor.

Instalarea Java Web Server Development Kit

Java Web Server Development Kit este utilizat pentru scrierea de servlet-uri. Este format dintr-un set de clase care completează biblioteca de clase din JDK şi din câteva utilitare. Au fost lansate recent câteva versiuni şi sunt deja planificate altele. Acest capitol studiază utilizarea versiunii 1.01 a JWSDK, care implementează versiunea 2.1 a servlet-ului API.

Sugestie de la LeeAnne.com Words to weave by

Mulţi dezvoltatori s-au încurcat în instrumentele cu nume asemănătoare puse la dispoziţie de Sun pentru lucrul în Java. Iată o referinţă rapidă:

- JSDK – Java Servlet Development Kit (kitul Java de dezvoltare de servlet-uri) este numele anterior al JWSDK.
- JWSDK – Java Web Server Development Kit (kitul de dezvoltare Java Web Server) este o bibliotecă de clase şi instrumente pentru dezvoltarea de servlet-uri şi de Java Server Pages.
- JDK – Java Development Kit (kitul de dezvoltare Java) este biblioteca de clase de bază şi compilatorul pentru Java.

Pentru dezvoltarea de servlet-uri veţi utiliza atât JDK, cât şi JWSDK.

Descărcaţi JWSDK 1.01 de la `http://java.sun.com/products/jsp/download.html` şi urmaţi instrucţiunile pentru instalarea sa. Asiguraţi-vă că setaţi calea dumneavoastră aşa cum este precizat în `readme.html` şi că aţi adăugat fişierul `servlets.jar` la calea de includere a claselor. Instrucţiunile din acest capitol presupun că nu aţi modificat setările prestabilite din `webserver.xml`.

Aşa e, fişierul de configurare pentru mini-serverul Web care vine cu JWSDK este un fişier XML. Începeţi să înţelegeţi larga răspândire (şi utilitatea) XML în domeniu.

Java Web Server Development Kit va fi înlocuit de Tomcat

Sun a anunţat că nu va mai furniza implementări de referinţă pentru Servlet API. Va păstra controlul lui API, dar codul care implementează efectiv API va fi scris de alţii. Sun suportă proiectul Tomcat de la Apache şi a desemnat Tomcat ca implementarea de referinţă oficială pentru versiunea 2.2 a specificaţiei Servlet şi pentru versiunea 1.1 a specificaţiei Java Server Pages. În momentul scrierii acestei cărţi, Tomcat este în versiune beta, aşa că servlet-urile din acest capitol sunt scrise cu JWSDK pentru versiunea 2.1 a servlet-ului API. Despre rolul jucat de Apache şi Sun în dezvoltarea servlet-urilor şi JSP puteţi citi mai multe la `http://java.sun.com/products/jsp/tomcat/`. Pentru a urmări proiectul Tomcat sau pentru a descărca o versiune beta, porniţi de la `http://jakarta.apache.org/tomcat/index.html`.

Un program CGI simplu care emite cod XML

Programele CGI sunt simplu de scris şi de executat de oriunde. Ele sunt apelate de obicei de serverul Web atunci când utilizatorul trimite un formular. Deşi au câteva dezavantaje care sunt discutate în secţiunile următoare, ele constituie o metodă simplă de a începe utilizarea XML pe server.

Executarea unui CGI Java prin intermediul unui înveliş Perl

Browserul trimite cereri pentru programele CGI într-un mod foarte asemănător cu cererile pentru pagini HTML simple. Puteţi configura serverul Web (de obicei prin marcarea extensiei fişierului cerut) astfel încât serverul să ştie că trebuie să execute o aplicaţie Java în loc să trimită fişierul înapoi. Totuşi, există câteva probleme referitoare la variabilele de mediu: informaţiile pe care serverul Web le primeşte de la browser sunt plasate în variabile de mediu, iar aplicaţia care rulează Java nu transferă în momentul execuţiei aceste variabile aplicaţiei dumneavoastră Java. Din acest motiv, şi pentru că unii dezvoltatori nu pot configura serverul lor Web astfel încât acesta să execute direct Java, cel mai avantajos este să scrieţi pentru aplicaţia dumneavoastră Java un înveliş Perl simplu, care setează variabilele utilizând opţiunea Java `-D` şi rulează aplicaţia. Listingul 17.1 prezintă unul care funcţionează foarte bine.

Listingul 17.1 Un înveliş Perl pentru o aplicaţie CGI Java

```
#print "Content-type: text/html\n\n";

$ARGV[0] =~ /.*[\\\/](.*?)\..*$/;
$class = $1;

#print "argv is #ARGV[0] - $class\n";

print `java -classpath .;c:/jdk1.1.6/lib/classes.zip;$ENV{'CLASSPATH'}
\"-DSERVER_SOFTWARE=$ENV{'SERVER_SOFTWARE'}\"
\"-DSERVER_NAME=$ENV{'SERVER_NAME'}\"
\"-DGATEWAY_INTERFACE=$ENV{'GATEWAY_INTERFACE'}\"
\"-DSERVER_PROTOCOL=$ENV{'SERVER_PROTOCOL'}\"
\"-DSERVER_PORT=$ENV{'SERVER_PORT'}\"
\"-DREQUEST_METHOD=$ENV{'REQUEST_METHOD'}\"
\"-DPATH_INFO=$ENV{'PATH_INFO'}\"
\"-DPATH_TRANSLATED=$ENV{'PATH_TRANSLATED'}\"
```

```
\"-DSCRIPT_NAME=$ENV{'SCRIPT_NAME'}\"
\"-DQUERY_STRING=$ENV{'QUERY_STRING'}\"
\"-DREMOTE_HOST=$ENV{'REMOTE_HOST'}\"
\"-DREMOTE_ADDR=$ENV{'REMOTE_ADDR'}\"
\"-DAUTH_TYPE=$ENV{'AUTH_TYPE'}\"
\"-DREMOTE_USER=$ENV{'REMOTE_USER'}\"
\"-DREMOTE_IDENT=$ENV{'REMOTE_IDENT'}\"
\"-DCONTENT_TYPE=$ENV{'CONTENT_TYPE'}\"
\"-DCONTENT_LENGTH=$ENV{'CONTENT_LENGTH'}\"
\"-DHTTP_ACCEPT=$ENV{'HTTP_ACCEPT'}\"
\"-DHTTP_USER_AGENT=$ENV{'HTTP_USER_AGENT'}\"
\"-DHTTP_REFERER=$ENV{'HTTP_REFERER'}\"
$class`;
```

Sugestie de la LeeAnne.com Words to weave by

Variabilele numite în acest cod (`SERVER_SOFTWARE`, `SERVER_NAME` şi aşa mai departe) sunt variabile de mediu CGI, accesibile de obicei din script-uri CGI, servlet-uri şi alte coduri server-side care sunt rulate la cererea serverului Web. Dacă nu sunteţi obişnuit cu aceste variabile de mediu particulare, liniştiţi-vă; ele nu sunt utilizate în acest capitol, care este despre XML, şi nu despre CGI. Dacă sunteţi curios, puteţi vedea ce înseamnă fiecare variabilă la `http://hoohoo.ncsa.uiuc.edu/cgi/env.html`.

Aspectul interesant al acestui înveliş este că funcţionează pentru fiecare clasă Java pe care o scrieţi. Îl puteţi utiliza pe servere Web care asociază extensiile de fişiere, cum ar fi `.cgi` sau `.class`, cu anumite acţiuni. Atât IIS cât şi PWS funcţionează în acest mod. Pe alte servere, este posibil să aveţi de scris un înveliş pentru fiecare program CGI Java şi să codaţi hard numele clasei Java în învelişul Perl. Liniile care fac ca acest înveliş să fie general sunt:

```
$ARGV[0] =~ /.*[\\\/](.*?)\..*$/;
$class = $1;
```

Dacă aţi cam uitat expresiile regulate Perl, iată o traducere a acelor linii. Funcţionează cu argumentul care a fost transferat învelişului Perl, iar acesta va fi o cale completă la clasa dumneavoastră Java. Împarte şirul în trei părţi: totul până la ultimul caracter / sau \, numele clasei dumneavoastră şi tot ce se află după ultimul caracter. (punct). Iată părţile expresiei regulate:

Primul `.*` înseamnă „totul până la".

`[\\\/]` înseamnă „fie un \, fie un /" (\ este un caracter escape deoarece / ar încheia de obicei expresia regulată). `\/` înseamnă „/", iar `\\` înseamnă „\".

`()` în `(.*?)` anunţă Perl să pună în `$1` orice se potriveşte cu această parte a expresiei.

`\..*?` înseamnă „." şi apoi orice altceva până la sfârşitul liniei.

Deci, dacă argumentul ar fi fost

```
C:\inetpub\wwwroot\cgi-bin\HelloWorld.class
```

fragmentul Perl de mai sus ar fi lăsat `HelloWorld` în variabila `$1`. Aceasta este salvată în `$class` (`$1` este o variabilă care poate fi suprascrisă uşor) şi utilizată în ultima linie a învelişului. Tot restul din acest înveliş formează o linie foarte lungă, împărţită prin caractere \. El rulează clasa dumneavoastră Java şi transferă şi toate variabilele de mediu CGI în cazul în care aveţi nevoie de ele.

Observaţi, de asemenea, că acest înveliş setează calea de includere a claselor la dosarul curent (.), la bibliotecile JDK şi la orice se află deja în calea de includere a claselor.

Configurarea serverului Web

Pentru a utiliza acest înveliş, puneţi mai întâi o copie a lui într-un loc util, cum este dosarul `cgi-bin` pentru serverul dumneavoastră Web. Apoi, configuraţi serverul dumneavoastră Web să îl apeleze ori de câte ori este cerută o resursă `.class`. Atât pentru IIS cât şi pentru PWS, urmaţi aceste instrucţiuni:

1. Afişaţi Microsoft Management Console (vezi Figura 17.3) alegând Start, Programs, Windows NT 4.0 Option Pack, Microsoft Personal Web Server (sau Microsoft Internet Information Server), Internet Service Manager.

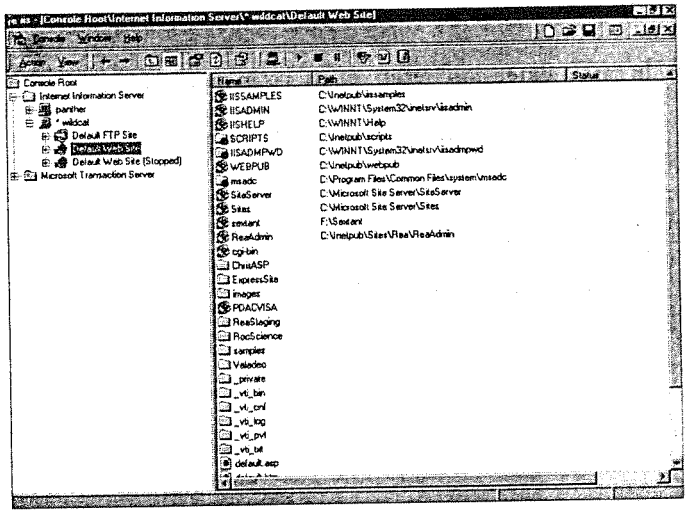


FIGURA 17.3
Configuraţi IIS sau PWS cu Microsoft Management Console.

2. Executaţi clic-dreapta pe Default Web Site şi alegeţi Properties. Selectaţi eticheta Home Directory (vezi Figura 17.4.)

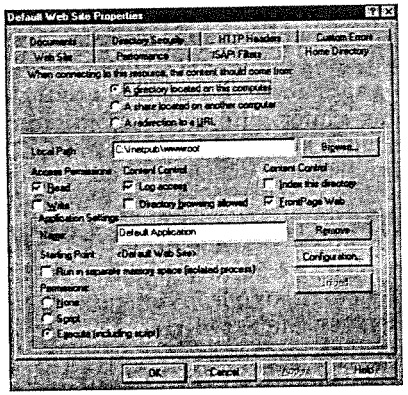
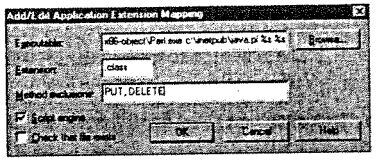


FIGURA 17.4
Eticheta Home Directory afişează opţiunile de configurare a fişierului.

3. Executați clic pe butonul Configuration din jumătatea de jos a ecranului. Se deschide caseta de dialog Application Configuration. Executați clic pe Add.
4. Se deschide caseta de dialog Add/Edit Application Extension Mapping (vezi Figura 17.5). În caseta Executable, introduceți calea completă la Perl, urmată de calea la învelișul Perl pentru CGI-ul Java prezentat anterior, și apoi șirul `%s %s`. De exemplu, pe un sistem, linia executabilă poate fi:

```
d:\Perl\5.00502\bin\MSWin32-x86-object\Perl.exe c:\inetpub\java.pl %s %s
```

FIGURA 17.5
Adăugați o mapare pentru extensia `.class`.



5. În caseta Extension treceți `.class`, iar în caseta Method Exclusions treceți `PUT, DELETE`. Acest lucru furnizează puțină securitate suplimentară, permițând utilizatorilor să execute cod pe serverul dumneavoastră, dar nu și să încarce sau să șteargă fișiere.
6. Executați clic pe OK în toate casetele de dialog până când vă întoarceți la consola de configurare.

Ajunși în acest punct, ar fi o idee bună să testați dacă funcționează învelișul și configurația serverului dumneavoastră Web. Găsiți (sau codați repede) o veche, dar bună aplicație Java, Hello World:

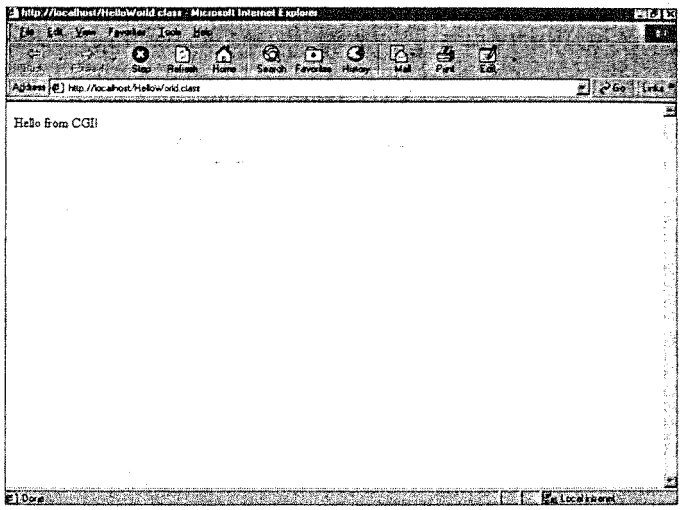
```
class HelloWorld
{
   public static void main(String[] args)
   {
      System.out.println("Content-Type: text/html\n");
      System.out.println("Hello from CGI!");
   }
}
```

Compilați această aplicație și copiați fișierul `.class` în rădăcina serverului dumneavoastră Web, probabil `c:\inetpub\wwwroot`. Acum, asigurați-vă că serverul dumneavoastră Web este pornit, încărcați-o într-un browser cu un URL cum ar fi `http://localhost/HelloWorld.class`. (În cazul în care copiați fișierul în alt loc și nu în rădăcină, atunci modificați URL-ul în consecință.) Ar trebui să vedeți ceva asemănător cu Figura 17.6, demonstrând că serverul Web a utilizat CGI pentru a apela învelișul Perl, care a rulat clasa Java, care a afișat un salut simplu, și că acel salut s-a întors prin învelișul Perl și serverul Web în browserul Web. Nu încercați să codați CGI XML în Java până când CGI-ul simplu Hello World nu funcționează corect.

Scrierea codului HTML care apelează CGI-ul

Un program CGI tipic este rulat ca rezultat al trimiterii de către utilizator a unui formular HTML. Există două moduri posibile în care poate fi trimis un formular: cu un `ACTION` al unui `GET` sau al unui `POST`. Multe programe CGI care au doar câțiva parametri utilizează `GET`, caz în care un URL este trimis browserului Web cu parametrii înglobați într-un URL obișnuit. Dacă intenționați să trimiteți mai mulți parametri, sau un parametru destul de lung, cum ar fi un șir XML, pentru a trimite formularul dumneavoastră este mai bine să folosiți metoda `POST`. `GET ACTION` nu se potrivește la șirurile lungi de date deoarece le trunchiază după un număr nedeterminat de caractere.

FIGURA 17.6
Înainte de a trece la o muncă mai complexă, testați un CGI simplu, scris în Java.



În programul CGI folosit ca exemplu în acest capitol, utilizatorului i se cere să introducă un șir XML scurt, care să arate astfel:

```
<name>kate</name>
```

Un nume este extras din acel cod XML și se trimite înapoi un cod XML ușor diferit. Evident, într-o aplicație reală, codul XML nu va fi introdus de utilizator. El va fi generat de un cod client-side inteligent, poate chiar ceva la fel de simplu ca un mic JavaScript care ar încadra șirurile introduse de utilizator într-un formular cu etichete de pornire și de încheiere. În acest capitol există un applet care furnizează o interfață utilizator mult mai inteligentă.

Iată un cod HTML care realizează un formular și execută un program CGI Java, numit ParseXML:

```
<HTML>
<HEAD><TITLE>Using XML</TITLE></HEAD>
<BODY>
<H1>XML from a form and CGI</H1>
<FORM ACTION="ParseXML.class" METHOD=POST>
  Enter some XML in the form &lt;name&gt;Your Name&lt;/name&gt;:
  <BR>
  <INPUT TYPE=text NAME=XML>
  <INPUT TYPE=submit NAME=submit VALUE="Process Request">
</FORM>
</BODY>
</HTML>
```

Observați că în acest formular, câmpul de intrare are `NAME=XML`. Acest nume este utilizat atunci când în aplicația CGI se extrag câmpurile formularului.

Acest formular poate fi văzut, așa cum este afișat în IE 5, în Figura 17.7. Este foarte simplu – doar o casetă de intrare și un buton de trimitere.

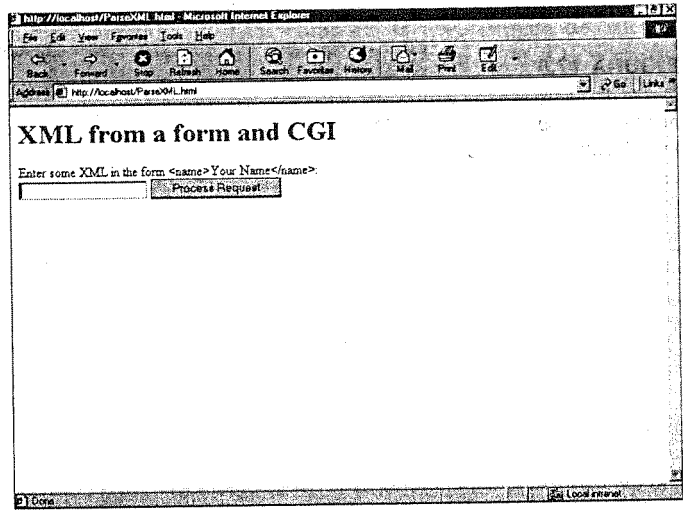


FIGURA 17.7
Formularul care adună parametrii pentru aplicația CGI este foarte simplu.

Scrierea codului CGI

Toate programele CGI au aceeași structură de bază:

Afișează tipul de conținut.
Extrag parametrii `GET` sau `POST`.
Utilizează parametrii pentru a efectua prelucrări.
Afișează o ieșire, ca rezultat al prelucrării.

Tipul de conținut

Tipul de conținut este important. Pentru majoritatea programelor CGI, acesta este `text/html`, dar pentru că acest program va emite cod XML, `text/plain` sau `text/xml` reprezintă o alegere mai bună. Dacă veți dori să afișați informații de depanare simple pe măsură ce dezvoltați, utilizarea `text/plain` elimină enervantele mesaje de eroare din IE 5 dacă ieșirea dumneavoastră de depanare nu formează un cod XML construit corect. (Consecințele tipului dumneavoastră de conținut sunt prezentate mai detaliat în secțiunea despre servlet-uri din acest capitol.)

Deci, prima linie a metodei `main()` din clasa `ParseXML` este

```
System.out.println("Content-Type: text/plain\n");
```

Extragerea parametrilor

Deoarece codul HTML care apelează acest program specifică `METHOD=POST`, parametrii (în acest caz șirul introdus de utilizator în caseta text de intrare) sunt disponibili la intrarea standard. Variabila de mediu `CONTENT_LENGTH`, care a fost transferată de învelișul Perl, vă spune câți octeți vă așteaptă. Îi puteți citi ca pe un tablou de octeți și să îi convertiți într-un șir Java:


```
String cl = System.getProperty("CONTENT_LENGTH");
System.out.println("got property");
if (cl != null && cl != "")
{
   int length = Integer.parseInt(cl);
   if (length > 0)
   {
      try
      {
         byte x[] = new byte[length];
         System.in.read(x);
         String encodedcontent = new String(x);
      }
      catch (Exception e)
      {
         System.out.println("<br>Error reading and parsing:" + e.toString());
      }
   }
}
```

Poate că vă gândiți că pasul următor este analizarea acelui șir, `encodedcontent`. Dar acesta nu se află într-o stare în care poate fi analizat. Dacă ați fi deschis într-un browser codul HTML prezentat anterior și ați fi introdus `<name>kate</name>` în caseta de text, apoi ați fi acționat butonul Process Request, atunci variabila `encodedcontent` ar fi conținut

```
XML=%3Cname%3Ekate%3C%2Fname%3E&submit=Process+Request
```

Șirul introdus de dumneavoastră a fost codat URL pentru a ascunde de serverul Web caractere ca < și >. Pentru a-l decoda, am scris o mică metodă bazată pe un utilitar care vine împreună cu pachetul de dezvoltare de servlet-uri, discutat în acest capitol.

Listingul 17.2 Inversarea codării URL

```
static private String URLDecode(String s)
   {
     StringBuffer sb = new StringBuffer();
     sb.setLength(0);
        for (int i = 0; i < s.length(); i++)
        {
          char c = s.charAt(i);
          switch (c)
          {
            case '+':
              sb.append(' ');
              break;
            case '%':
              try
              {
                sb.append((char) Integer.parseInt(s.substring(i+1, i+3),16));
                i += 2;
              }
              catch (NumberFormatException e)
              {
                throw new IllegalArgumentException();
              }
```

```
            catch (StringIndexOutOfBoundsException e)
            {
              String rest  = s.substring(i);
              sb.append(rest);
              if (rest.length()==2)
              i++;
             }

              break;
           default:
              sb.append(c);
              break;
          }
       }
       return sb.toString();
  }
```

Utilizarea acestei metode pentru a decoda conținutul este simplă:

```
String content = URLDecode(encodedcontent);
```

Presupunând că în caseta de text a fost introdus același nume ca și mai înainte, șirul conținut păstrează acum

```
XML=<name>kate</name>&submit=Process Request
```

Operația care a mai rămas este extragerea codului XML, ignorând informația despre butonul din formular care a fost acționat. Iată o abordare care nu este deloc generală:

```
int start = content.indexOf("XML=")+4;
int end = content.indexOf("&",start);
xml= content.substring(start,end);
```

Într-o aplicație CGI reală, veți utiliza probabil un tabel hash sau o structură de date asemănătoare, veți analiza deodată întreg conținutul și apoi veți extrage parametrii doriți. Dar după cum probabil v-ați dat seama până acum, în Java nu sunt scrise prea multe aplicații CGI reale. Această abordare subșir de mântuială funcționează pentru scopurile acestui capitol.

Utilizarea parametrilor

Pentru a analiza șirul XML, creați un analizor și transferați-i un `StringReader` format din șir:

```
DOMParser p = new DOMParser();
p.parse(new InputSource(new StringReader(xml)));
Document in = p.getDocument();
```

Acest cod utilizează un analizor DOM – pentru a seta un analizor SAX veți face ceva asemănător. Programul CGI din exemplu se așteaptă să primească un element `<name>`. Iată codul care găsește acel element și primește valoarea nodului său text:

```
NodeList nodes = in.getElementsByTagName("name");
if(nodes.getLength() > 0)
{
   String submittedname = nodes.item(0).getChildNodes().item(0).getNodeValue();
}
```


Sugestie de la LeeAnne.com Words to weave by — Dacă aveți dubii în legătură cu această linie de cod DOM, poate veți dori să recitiți Capitolul 10, „DOM (Document Object Model) – Modelul document obiect".

Se așteaptă ca programul CGI folosit ca exemplu să emită un cod XML care să arate astfel:

```
<person>
  <name>
    <first>Kate</first>
    <last>Gregory</last>
  </name>
</person>
```

Într-o aplicație reală, codul HTML emis va fi bazat pe valorile găsite într-o bază de date sau calculate într-un anumit fel de la o altă intrare. În această aplicație exemplu, codul XML simplu codat hard este suficient pentru a demonstra aceste tehnici. Pentru a fi mai interesant, un prenume primește un tratament special:

```
String lastname = "";
if (submittedname.equals("Kate")) lastname = "Gregory";
```

Iată codul Java simplu care creează codul XML ce va fi afișat:

```
TXDocument out = new TXDocument();

Element person = out.createElement("person");
out.appendChild(person);

Element name = out.createElement("name");
person.appendChild(name);

Element first = out.createElement("first");
name.appendChild(first);
first.appendChild(out.createTextNode(submittedname));

Element last = out.createElement("last");
name.appendChild(last);
String lastname = "";
if (submittedname.equals("Kate")) lastname = "Gregory";

last.appendChild(out.createTextNode(lastname));
out.printWithFormat(System.out);
```

Acest cod utilizează un `TXDocument` deoarece apelul metodei `printWithFormat()` este un mod foarte comod de a afișa un bloc de cod XML frumos formatat. DOM nu furnizează o metodă pentru afișarea unui întreg arbore. XML4J conține un exemplu numit `DOMWriter`, dar acesta scrie în `System.out`, și nu la ieșirea pe care ați alege-o dumneavoastră. Utilizarea lui `TXDocument` (care nu există în biblioteca fiecărui analizor) face din acest lucru un simplu apel de metodă.

Probleme cu CGI

Aproape nimeni nu scrie efectiv aplicații CGI Java. După cum ați văzut, pentru a transfera variabile de mediu CGI-ului dumneavoastră, trebuie să utilizați un înveliș Perl. De

asemenea, dumneavoastră trebuie să decodați și să extrageți parametrii. Aceste probleme vă pot supăra în timpul scrierii primului dumneavoastră CGI Java, dar după aceea vă puteți baza pe o bibliotecă a propriei dumneavoastră munci pentru a simplifica operațiile.

O problemă mai mare cu CGI este aceea că nu se scalează bine. O abordare CGI care implică Perl (chiar și acest exemplu Java simplu implică un înveliș Perl) scalează chiar mai slab. Pentru a înțelege de ce, gândiți-vă ce se întâmplă dacă doi sau mai mulți utilizatori trimit simultan același formular (sau rulează într-o anumită manieră aceeași resursă CGI).

Serverul Web pornește o copie completă a programului CGI pentru cererea fiecărui utilizator. Fiecare program are propriile variabile de mediu și spațiu de memorie. Dacă este implicat și Perl, fiecare program rulează și o copie a interpretorului Perl. Aceasta reprezintă un consum imens de resurse. În ciuda faptului că un server poate manevra cu ușurință două (sau chiar 10) cereri simultane ale aceleiași resurse CGI, există o încărcătură insuportabilă dacă trebuie satisfăcute sute sau mii de cereri. De asemenea, chiar și la numai câteva cereri există o întârziere pentru utilizator până când sunt pornite aplicația CGI și interpretoarele asociate.

O abordare mult mai eficientă (și mai scalabilă) ar fi aceea ca un proces să ruleze continuu, iar fiecare cerere care intră să fie manevrată de un nou fir din acel proces. Utilizatorii finali vor primi răspunsuri mai rapide, iar serverul utilizează mult mai puțină memorie și timp de execuție. Exact așa funcționează servlet-urile.

O abordare servlet

Dacă sunteți obișnuit cu applet-uri și cu CGI, scrierea de servlet-uri este simplă. (Dacă nu sunteți obișnuit cu CGI, învățarea servlet-urilor este mult mai ușoară decât învățarea CGI, în special atunci când doriți să lucrați în Java. Așa cum ați văzut, lucrul cu CGI în Java este destul de dificil.)

Pentru a scrie un servlet, scrieți o clasă care extinde o clasă bibliotecă numită `javax.servlet.http.HttpServlet`. (Această clasă face parte dintr-o bibliotecă de clase suplimentară care vine cu Java Web Server Development Kit.) Seamănă foarte mult cu modul în care un applet este o clasă care extinde `java.applet.Applet`.

Atunci când scrieți un applet, suprascrieți metodele din clasa superioară, cum este `paint()`. Atunci când scrieți un servlet, suprascrieți `doGet()` și `doPost()`. Acestea sunt metodele care sunt apelate ori de câte ori un browser trimite o metodă `GET` sau `POST` serverului Web pentru un servlet. O cerere care vine de la o legătură sau de la un utilizator care introduce URL-ul pentru un servlet este întotdeauna o cerere `GET`. O cerere rezultată din trimiterea unui formular poate fi `GET` sau `POST`, în funcție de atributul `METHOD` din eticheta `<FORM>` din codul HTML.

Structura unui servlet

Un servlet tipic arată astfel:

```
import javax.servlet.*;
import javax.servlet.http.*;
public class SampleServlet extends HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException
  {
    // write form HTML to res
  }
public void doPost(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException
  {
    // get form fields from req, write output HTML to res
  }
}
```

Pentru a dezvolta un servlet, parcurgeți pașii următori:

1. Codați servlet-ul. Deoarece este o clasă publică, asigurați-vă că numele fișierului se potrivește cu numele clasei.
2. Compilați servlet-ul.
3. Copiați fișierul `.class` în dosarul în care serverul Web caută servlet-urile. În mod prestabilit, Java Web Server caută în dosarul `servlets` din dosarul de instalare.
4. Introduceți în browser URL-ul la servlet. Prin aceasta se testează metoda dumneavoastră `doGet()`. De exemplu, dacă servlet-ul dumneavoastră se numește `SampleServlet` și l-ați copiat în dosarul în care Java Web Server caută servlet-urile, atunci URL-ul este `http://localhost:8080/servlet/SampleServlet`.

Sugestie de la LeeAnne.com Words to weave by — Șirul `servlet` din URL este mapat la dosarul `servlets` din dosarul de instalare. În URL nu introduceți `servlets`.

5. Dacă metoda dumneavoastră `doGet()` emite un formular, atunci completați câmpurile și trimiteți formularul. Aceasta testează metoda dumneavoastră `doPost()`.

Emiterea unui formular simplu

În servlet-ul folosit ca exemplu în acest capitol, utilizatorului i se cere să introducă un șir XML scurt care să arate astfel:

```
<name>kate</name>
```

Din acel cod XML este extras un nume și se returnează un cod XML ușor diferit. Iată codul pentru metoda `doGet()` care emite formularul:

```
public void doGet(HttpServletRequest req, HttpServletResponse res)
                throws ServletException, IOException
  {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML><HEAD><TITLE>Using XML</TITLE></HEAD>");
    out.println("<BODY><H1>XML from a form</H1>");
    String Url = req.getServletPath();
    out.println("<FORM METHOD=\"POST\" ACTION=\"" + Url + "\">");
    out.println("Enter some XML in the form &lt;name&gt;Your Name&lt;/name&gt;:");
    out.println("<br> <INPUT TYPE=\"TEXT\" NAME=\"XML\" size=\"50\"><BR>");
    out.println("<INPUT TYPE=submit VALUE=\"Process Request\">");
    out.println("</FORM></BODY></HTML>");
    out.close();
  }
```

Acest cod setează tipul de conținut pentru documentul emis la text HTML, primește un PrintWriter care este asociat documentului emis și apoi scrie în el codul HTML. Întrebând obiectul de cerere despre calea la acest servlet, codul asigură că aceeași clasă de servlet-uri care a manevrat cererea GET manevrează și cererea POST. Codarea hard a numelui servlet-ului dumneavoastră în câmpul ACTION va duce la un cod greu de întreținut și la erori greu de găsit atunci când copiați pentru prima dată codul vechi într-un proiect nou.

Observați că în acest formular, câmpul de intrare are NAME=XML. Acest nume este utilizat atunci când în metoda doPost() se extrag câmpurile formularului.

Prelucrarea formularului

Prelucrarea unui formular se compune din două părți. Una dintre ele este extragerea câmpurilor din formular și cealaltă este executarea unor acțiuni asupra acelor câmpuri. În servlet-ul exemplu există un singur câmp de interes. El poate fi extras astfel:

```
String xml;

String[] vals = req.getParameterValues("XML");
if (vals != null)
{
   xml = vals[0];
}
```

Observați că șirul transferat lui getParameterValues(), „XML", este numele etichetei INPUT din eticheta FORM emisă de doGet().

Pentru a analiza șirul, creați un analizor și transferați-i un StringReader realizat din acest șir:

```
DOMParser p = new DOMParser();
p.parse(new InputSource(new StringReader(xml)));
Document in = p.getDocument();
```

Acest cod seamănă exact cu codul Java din aplicația CGI prezentată anterior. Deci este codul care găsește acel element și primește valoarea nodului său text:

```
NodeList nodes = in.getElementsByTagName("name");
if(nodes.getLength() > 0)
{
   String submittedname = nodes.item(0).getChildNodes().item(0).getNodeValue();
}
```

Se așteaptă ca servlet-ul exemplu să emită același cod XML ca exemplul CGI. El seamănă cu acesta:

```
<person>
  <name>
    <first>Kate</first>
    <last>Gregory</last>
  </name>
</person>
```

Așa cum s-a menționat anterior, într-o aplicație reală, codul HTML emis se va baza pe valorile găsite într-o bază de date sau calculate într-un alt mod dintr-o altă intrare.

Iată codul Java simplu care creează codul XML ce va fi afișat:


```
TXDocument out = new TXDocument();

Element person = out.createElement("person");
out.appendChild(person);

Element name = out.createElement("name");
person.appendChild(name);

Element first = out.createElement("first");
name.appendChild(first);
first.appendChild(out.createTextNode(submittedname));

Element last = out.createElement("last");
name.appendChild(last);
String lastname = "";
if (submittedname.equals("Kate")) lastname = "Gregory";

last.appendChild(out.createTextNode(lastname));
out.printWithFormat(pw);
```

Acest cod utilizează un TXDocument, așa cum a făcut și codul CGI, și tot din aceleași motive.

Tipul conținutului este important

Figura 17.8 prezintă ieșirea acestui servlet în IE 5.

Dacă ați încărcat XML în IE 5, atunci știți că de obicei el este afișat în format arborescent. Motivul pentru care arată atât de monoton în Figura 17.8 este că tipul de conținut era setat la text/plain. Modificarea tipului de conținut la text/xml va produce ceea ce vedeți în Figura 17.9. IE formatează diferit etichetele și conținutul lor și vă permite să extindeți sau să restrângeți etichetele ușor și repede.

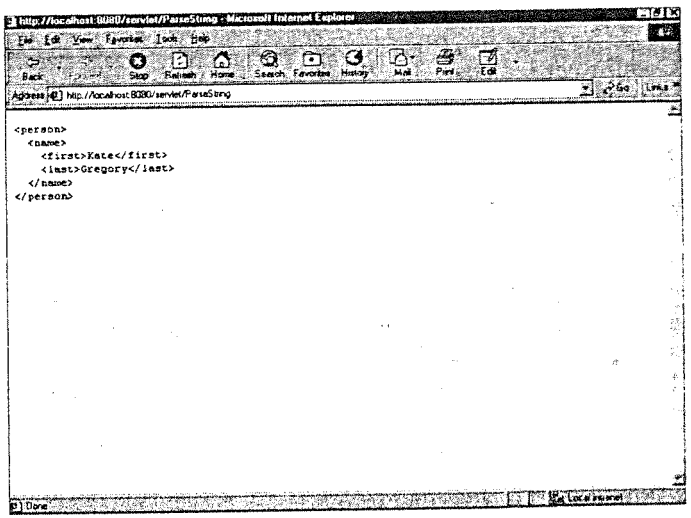


FIGURA 17.8
Servlet-ul exemplu din acest capitol emite cod XML. Aici, IE 5 îl tratează ca text simplu.

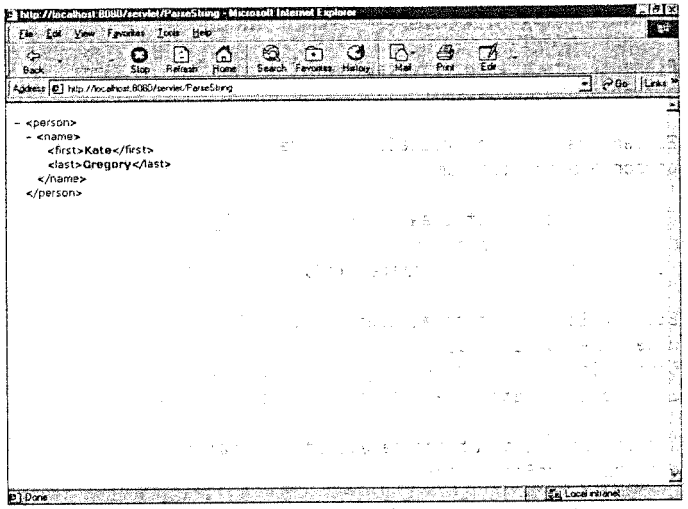


FIGURA 17.9
Atunci când IE 5 ştie că ieşirea servlet-ului este XML, dumneavoastră obţineţi o afişare arborescentă mult mai utilă.

Metoda `doGet()` din acest servlet setează tipul de conţinut la `text/html`. Figura 17.10 prezintă ieşirea în IE 5, împreună cu fereastra View Source. Vedeţi ce s-a întâmplat? Browserul ignoră toate etichetele, presupunând că acestea nu fac parte din HTML. Nu uitaţi niciodată să setaţi tipul dumneavoastră de conţinut la valoarea adecvată.

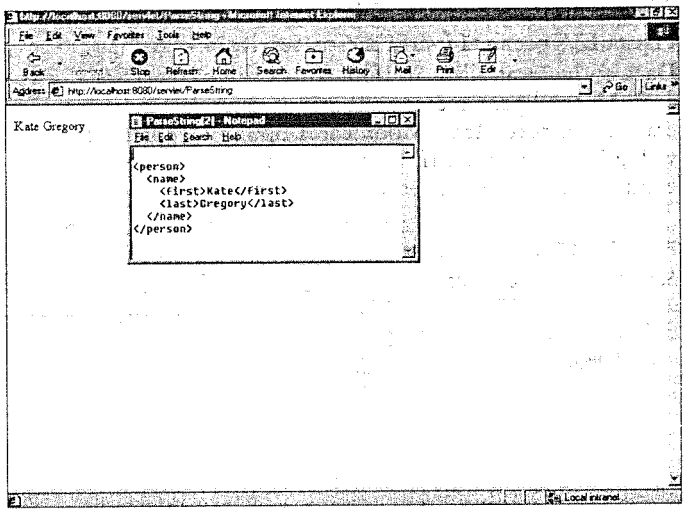


FIGURA 17.10
Atunci când IE 5 consideră că ieşirea servlet-ului este HTML, toate etichetele dispar. Ele mai există însă în sursa paginii.

Codul XML afişat într-un browser nu este o privelişte prea plăcută. Este o metodă bună de a testa dacă este emis codul XML pe care îl doriţi, dar nu este nici pe departe un produs terminat. Lista următoare explică cele trei lucruri pe care le puteţi face în loc să trimiteţi cod XML brut unui browser care nu bănuieşte nimic:

1. Puteţi presupune că browserul este IE 5 şi să adăugaţi o foaie de stil referinţă, astfel încât browserul să redea corect documentul XML. Aşa cum aţi învăţat în Capitolul 14, „Utilizarea generaţiei următoare de foi de stil extensibile – XSL", aceasta nu este o simplă presupunere, iar XSL sau XSLT care funcţionează în IE 5 nu funcţionează întotdeauna în alte contexte.
2. Puteţi utiliza XSL pentru a transforma codul dumneavoastră XML în HTML pe server şi să trimiteţi browserului cod HTML. Deci, indiferent de browserul folosit de utilizator, informaţia dumneavoastră este afişată corect.
3. Puteţi scrie un applet sau un alt cod inteligent client-side care să ceară cod XML şi să îl utilizeze într-o manieră inteligentă pentru a arăta informaţia utilizatorului final. Acest lucru cere un browser care suportă applet-uri, o presupunere destul de sigură acum.

Transformarea pe server a codului XML în HTML

Puteţi transforma pe server codul dumneavoastră XML în cod HTML care funcţionează în orice browser. XT-ul lui James Clark este un produs bun de utilizat în acest scop. Puteţi obţine o copie de la `http://www.jclark.com/xml/xt.html`. Dezarhivaţi fişierul într-un dosar potrivit şi nu uitaţi să adăugaţi `xt.jar` la calea de includere a claselor dumneavoastră, care probabil că este deja destul de lungă.

În `xt.jar` există câteva clase Java utile pentru transformarea codului XML într-un alt fel de XML sau în HTML, cu ajutorul unei foi de stil. Modificările aduse codului, prezentate anterior în acest capitol, sunt destul de mici, în special dacă utilizaţi varianta DOM a translatorului şi profitaţi de faptul că aţi creat deja un arbore DOM.

Lucrul cu clasele XT este puţin mai uşor dacă documentul DOM de ieşire care este construit ca ieşire este o instanţă a `com.sun.xml.tree.XmlDocument` decât un `TXDocument`. Comoda metodă `printWithFormat()` a lui `TXDocument` poate fi înlocuită cu metoda `write()` a lui `XmlDocument`. Trecerea la `XmlDocument` nu va avea nici o altă consecinţă în codul prezentat în acest capitol. Totuşi, clasa nu există în toate pachetele pe care le-aţi descărcat şi instalat până acum. Pentru a obţine clasa `XmlDocument`, descărcaţi API-ul Java pentru pachetul XML Parsing de la `http://java.sun.com/xml/download.html`. (Poate va trebui să vă înregistraţi şi să vă logaţi pentru a primi pachetul.) Extrageţi clasele şi adăugaţi `jaxp.jar` la calea de includere a claselor dumneavoastră.

Presupunând că aveţi un `XmlDocument` numit `out` şi un `PrintWriter` numit `pw`, iată cum puteţi transforma acel document utilizând o foaie de stil scrisă anterior:

```
XmlDocument transformed = new XmlDocument();
new XSLTransformEngine().
createTransform(XmlDocument.createXmlDocument("http://localhost:8080/name.xsl")).
➥transform(out,transformed);
transformed.write(pw);
```

Aceste trei linii de cod, destul de diforme, fac destul de multe:

Creează un nou document XML care să păstreze arborele transformat.
Creează un motor de transformare.
Creează un al treilea document XML, transferând un URL al unei foi de stil.
Realizează o transformare bazată pe acest document foaie de stil.
Utilizează transformarea pentru a modifica documentul original în documentul transformat.
Afişează documentul transformat.

Linia din mijloc este o singură linie, deoarece nu este nevoie să păstraţi referinţe la un

obiect intermediar, cum ar fi documentul foaie de stil sau obiectul transformat. Creați-l, utilizați-l și aruncați-l la gunoi.

Notă Codul anterior funcționează la fel de bine într-o aplicație CGI Java sau într-un servlet.

Iată codul XSL care însoțește exemplul XML utilizat pe parcursul acestui capitol:

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xt="http://www.jclark.com/xt"
               version="1.0"
                extension-element-prefixes="xt">
<xsl:template match="/">
  <HTML><HEAD><TITLE>Name from XML</TITLE></HEAD><BODY>
  <xsl:apply-templates select="person/name" />
  </BODY></HTML>
</xsl:template>
<xsl:template match="name">
  <h1>The name is:</h1>
  <xsl:value-of select="first"/>
  <i><xsl:value-of select="last"/></i>
</xsl:template>
</xsl:stylesheet>
```

Am modificat servlet-ul exemplu pentru acest capitol adăugându-i cele trei linii de cod pentru a utiliza `name.xsl` și a emite, în loc de XML, cod HTML. Figura 17.11 prezintă aspectul codului HTML emis în browser, împreună cu sursa. După cum puteți vedea, în sursa trimisă browserului nu există nici un semn din codul XML original. Utilizatorul final nu poate să își dea seama că sistemul dumneavoastră utilizează XML.

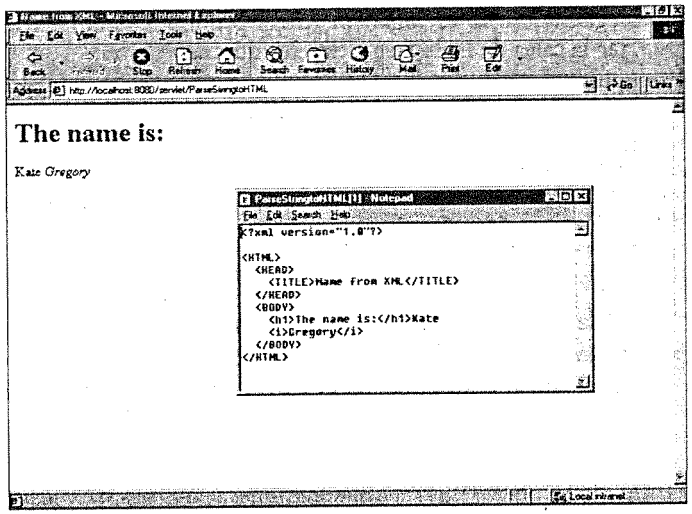


FIGURA 17.11
Atunci când pe server transformați cod XML în cod HTML, browserul nu vede niciodată codul XML original.

Acest servlet depune mai multă muncă decât este nevoie prin efectuarea transformărilor pentru fiecare caz. Altfel, puteți utiliza informația browserului în cererea servlet-ului pentru a detecta browserul care este utilizat. Dacă acesta este IE 5 sau mai recent, trimiteți codul XML cu o etichetă foaie de stil și lăsați browserul să facă treaba. Dacă nu, transformați-l dumneavoastră. Ați putea chiar utiliza foi de stil diferite pentru browsere diferite, deci utilizatorii care vizualizează situl dumneavoastră pe un telefon celular primesc pagini lizibile, asemenea dezvoltatorilor care au monitoare de 21" setate la rezoluții de 1600×1200.

Deoarece servlet-urile pot fi înlănțuite, vă puteți gândi la scrierea unui servlet înlănțuit care interceptează toate cererile XML și aplică pe loc foile de stil potrivite, returnând browserului cod HTML curat. Acest lucru va face situl dumneavoastră mult mai accesibil pentru utilizatorii cu browsere care nu au suport XML, făcându-le viața ușoară celor care construiesc pagini pe situl dumneavoastră. Dacă performanța este o problemă (și, mai devreme sau mai târziu, performanța va fi o problemă), păstrați în cache versiuni populare de transformări pentru fișiere populare și returnați-le rapid. În acest caz va trebui să generați transformări noi numai atunci când primiți o cerere neobișnuită.

BIBL. CENTR. UNIV. „M. EMINESCU" IAȘI INFORMATICĂ

Un applet care cere cod XML

Dacă nu vă puteți baza pe cunoștințele browserului în ceea ce privește codul XML, are oare sens să emiteți cod XML brut dintr-un cod server-side? În mod sigur că are – atunci când acest cod nu este cerut de un browser.

Servlet-ul exemplu prezentat deja în acest capitol are propria interfață utilizator: un formular HTML emis de servlet ca răspuns la o cerere `GET`. Puteți realiza o interfață utilizator mult mai atrăgătoare prin scrierea unui applet care primește numele și trimite la ieșire codul pentru utilizator. El poate primi codul XML rulând un servlet care generează cod XML. Acest servlet seamănă cu servlet-ul `ParseString`, dar emite cod XML ca răspuns la o cerere `GET`. Se întâmplă așa, în parte deoarece nu trebuie emis un formular ca răspuns la o cerere `GET` atunci când applet-ul furnizează interfața, și în parte deoarece trimiterea unei cereri `POST` dintr-un applet este mai dificilă decât trimiterea unei cereri `GET`.

De fapt, primirea codului dumneavoastră XML de la server nu este deloc o muncă suplimentară. Pe server aveați un șir XML pe care doreați să îl citiți, deci după crearea unui analizor, apelați astfel metoda lui de analizare:

```
p.parse(new InputSource(new StringReader(xml)));
```

Atunci când codul XML pe care doriți să îl citiți este localizat la un URL (sau este ieșirea unui servlet sau program CGI care poate fi adresat printr-un URL), trebuie numai să transferați URL-ul (ca șir) metodei de analizare, astfel:

```
p.parse(url);
```

Este efectiv mult mai ușor decât analizarea dintr-un șir. Analizorul face tot ce trebuie făcut, realizând o conexiune HTTP, citind tot textul trimis prin conexiune și așa mai departe.

Având acestea în minte, singura parte dificilă în scrierea unui applet inteligent este designul interfeței utilizator. În privința acestui lucru voi fi puțin zgârcită – subiectul acestui exemplu nu este realizarea de applet-uri bune. Un câmp text, `TextField`, pentru introducerea numelui, un buton, `Button`, prin care să cerem să fie trimis serverului, și niște spații libere pentru a afișa răspunsul serverului, sunt toate cele necesare. Metoda `init()` a applet-ului organizează totul:

```
public class GetXML extends Applet implements ActionListener
{
  private String fullname;
  private TextField name;
  private Button button;
  public void init()
  {
     add(new Label("Enter Your Name: "));
     name = new TextField(30);
        add(name);
        button = new Button("Process Request");
        add(button);
        button.addActionListener(this);
  }
  public void actionPerformed(ActionEvent ev)
  {
     // to be shown shortly
  }
  public void paint(Graphics g)
  {
     g.drawString(fullname,10,100);
  }
}
```

Acest cod ar trebui să pară familiar celor care au scris applet-uri cu AWT. Atunci când utilizatorul execută clic pe buton, este apelată metoda `actionPerformed()`. Această metodă construiește un URL pentru servlet (în funcție de locația applet-ului), analizează conținutul URL-ului, extrage prenumele și numele și construiește șirul `fullname`. Iată cum arată:

```
public void actionPerformed(ActionEvent ev)
{
     try
     {
     URL xml = new URL(getDocumentBase(),"servlet/ServeXML?XML=<name>" +
name.getText() + "</name>");
        DOMParser p = new DOMParser();
        p.parse(xml.toString());
        Document response = p.getDocument();

     String firstname="";
     String lastname="";
        NodeList nodes = response.getElementsByTagName("first");
        if(nodes.getLength() > 0)
        {
            firstname = nodes.item(0).getChildNodes().item(0).getNodeValue();
     }
        nodes = response.getElementsByTagName("last");
        if(nodes.getLength() > 0)
        {
            lastname = nodes.item(0).getChildNodes().item(0).getNodeValue();
     }
     fullname = firstname + " " + lastname;
     repaint();
```



```
     }
     catch (Exception e)
     {
   System.out.println("Exception caught: " +e.toString());
     }
  }
```

Întregul cod ar trebui să vi se pară cunoscut din exemplele CGI și servlet care găsesc elementul `<name>` și extrag valoarea. Figura 17.12 prezintă applet-ul după introducerea unui nume și acționarea butonului.

URL-ul numește un servlet, `ServeXML`, care seamănă foarte mult cu `ParseXML`, singura excepție fiind aceea că returnează codul XML din metoda `doGet()`, și nu din `doPost()`. De fapt, am făcut o copie a lui `ParseXML`, am șters `doGet()` și am redenumit `doPost()` în `doGet()`. Nu este nevoie să prezentăm aici acest cod.

Parametrii sunt încorporați chiar într-un URL, cu o cerere `GET` separată de numele de fișier printr-un caracter `?`. Dacă numele poate să conțină spații, ghilimele sau alte caractere neobișnuite, atunci utilizați un `URLEncoder` pentru a codifica parametrii înainte de trimiterea lor la server. Programele CGI trebuie să decodifice parametrii lor, iar servlet-urile își fac treaba prin obiectul `HttpRequest`.

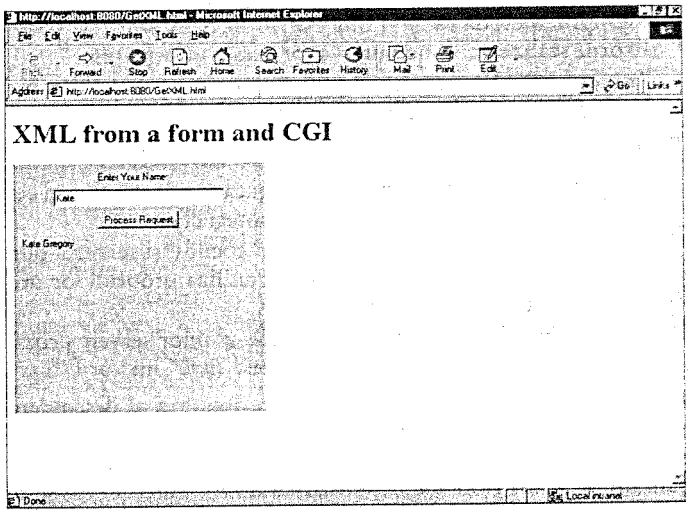


FIGURA 17.12
Acest applet transformă intrarea obișnuită în cod XML pe care îl trimite unui servlet. Servlet-ul trimite codul XML ca răspuns, iar applet-ul îl utilizează pentru a afișa un șir.

Pentru a afișa applet-ul, trebuie să încărcați un cod HTML în browser. Browserul este posibil să se afle pe un calculator care nu are instalat XML4J, deci dumneavoastră trebuie să arătați de unde poate fi acesta preluat, utilizând atributul `ARCHIVE` al etichetei `APPLET`. Asigurați-vă că ați copiat fișierul `jar` în același dosar în care se află applet-ul. Iată o pagină foarte simplă care realizează aceste obiective:

```
<HTML>
<HEAD><TITLE>Using XML with an applet</TITLE></HEAD>
<BODY>
<H1>XML from a form and CGI</H1>
<APPLET CODE="GetXML.class" HEIGHT=300 WIDTH=300 ARCHIVE="xml4j.jar">
</APPLET>
</BODY>
</HTML>
```

Avem aici tot ce este necesar. Applet-ul și servlet-ul formează împreună un sistem XML bun și comunică între ele folosind XML. În acest exemplu, ambele utilizează DOM; de fapt, ambele utilizează XML4J, dar nu este nevoie să se procedeze astfel. Ar fi destul de simplu să avem componenta server scrisă într-un alt limbaj decât Java, să utilizăm un analizor complet diferit, dar totuși să scriem cod XML pe care applet-ul să știe cum să îl utilizeze. Aceasta este adevărata puterea a lui XML.

Utilizarea soclurilor în locul HTTP

Acest capitol v-a arătat cum să utilizați, prin HTTP, XML pe server. Componenta server-side poate fi un program CGI sau, dacă aveți un server Web adecvat, un servlet. Programul client-side poate fi un browser Web, un applet sau orice altceva capabil să primească un flux de conținut de la un URL. De cele mai multe ori, utilizarea HTTP pentru a face schimb de informații cu serverul este calea de urmat. Printre avantaje se numără:

- O varietate de soluții server-side diferite ușurează „interceptarea" cererilor HTPP și rutarea lor la cod executabil. Programele CGI și servlet-urile, de exemplu, sunt destul de ușor de scris.
- O diversitate de biblioteci client-side ușurează cererea de informații prin HTTP. De exemplu, metoda `parse()` din API-urile SAX și COM poate primi un URL, iar analizorul realizează conexiunea și primește informațiile.
- Traficul HTTP este permis prin majoritatea protecțiilor firewall, fără a fi nevoie de înțelegeri speciale cu administratorii sistemului dumneavoastră.

Totuși, HTTP nu este întotdeauna soluția potrivită. Unele dintre dezavantaje sunt:

- Nu puteți transfera șiruri foarte lungi ca parametri ai unei cereri `GET` – iar cererile `POST` sunt dificil de trimis de la acele biblioteci client-side.
- HTTP este un protocol nesigur, deci clientul și serverul dumneavoastră nu pot avea o conversație continuă ca și când ar utiliza propriul lor protocol și ar păstra soclul dintre ei întotdeauna deschis.
- Cererile HTTP pot fi stocate în cache-ul unor servere proxy între client și server, iar atunci când cache-ul este defect, datele „live" nu vor fi foarte recente.

Din aceste motive, pe lângă altele, poate veți dori să deschideți o conexiune soclu între un applet și o aplicație care rulează pe server. Dacă sunteți obișnuit cu programarea de socluri, adăugarea XML la această combinație nu este deloc o problemă. Faceți un simplu schimb de șiruri prin soclu – șiruri de XML. Puteți citi și scrie acele șiruri așa cum ați văzut deja în celălalt cod exemplu din acest capitol.

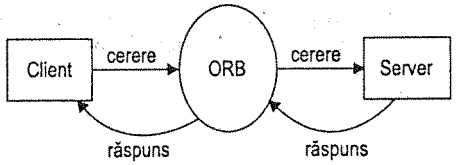
CORBA

CORBA (Common Object Request Broker Architecture), arhitectura brokerului de cerere de obiecte comune, este o arhitectură de componente care simplifică dezvoltarea client-server sau pentru legături multiple. Componentele pot fi pe același calculator sau distribuite într-o rețea. Un client CORBA face o cerere unui server CORBA printr-un ORB, adică Object Request Broker (broker de cerere de obiecte).

Acest ORB efectuează toată munca necesară, care vă este prezentată în lista următoare:

- Găsirea serverului (care poate fi pe alt calculator)
- Activarea serverului
- Colectarea parametrilor care vor fi transferați metodei (marshalling)
- Distribuirea parametrilor returnați de metodă (unmarshalling)

FIGURA 17.13
Între clienți și servere se află un broker de cerere de obiecte.



Toate aceste operații pot fi realizate de ORB dacă dispune de o referință de obiecte interoperabilă (Interoperable Object Reference), IOR. Un IOR generat de un ORB poate fi folosit de un alt ORB – de aici vine interoperabilitatea. Majoritatea acestor brokeri oferă un pachet de servicii înrudite pentru a ușura generarea unui IOR pentru un server, cum ar fi Naming Service, care identifică serverele după nume.

CORBA nu este dependent de nici un limbaj. Definiți serviciile pe care le oferă serverele dumneavoastră (cererile pe care le ascultă) folosind o notație independentă de limbaj, numită limbaj de definire a interfeței (Interface Definition Language), IDL. Utilitățile furnizate de ORB generează din acest IDL cod Java, cod C++ sau cod în orice limbaj doriți.

Sugestie de la LeeAnne.com Words to weave by

IDL-ul CORBA nu este unic. De exemplu, definițiile de interfețe COM sunt de asemenea descrise ca IDL, dar cele două notații nu sunt identice.

CORBA ușurează conlucrarea între obiecte scrise în limbaje diferite. Atât timp cât obiectul care cere servicii poate contacta un ORB și atât timp cât obiectul care furnizează acele servicii poate fi contactat de un ORB, cele două obiecte pot conlucra. Ordinea parametrilor, tipurile de bază suportate de limbaj și alte probleme similare nu contează pentru un dezvoltator CORBA. ORB-ul realizează toată munca în locul dumneavoastră. Un client scris în Java poate trimite cereri unui server scris în C fără ca dumneavoastră să fiți preocupat de JNI sau de alte tehnologii înrudite. Analog, obiectele dumneavoastră pot să se conecteze printr-o rețea fără să folosească RMI. ORB-ul face tot.

CORBA, Java și XML au multe în comun. Numeroși furnizori au dezvoltat ORB-uri la care accesul poate fi la fel de simplu ca și apelul unei clase publice din Java. Toate trei sunt soluții pentru toate platformele. Un slogan ar putea fi „date portabile, cod portabil". XML furnizează datele portabile, deoarece formatul XML poate fi înțeles pe orice platformă. Java este codul portabil. Iar CORBA este liantul care simplifică apelarea acelui cod de oriunde, chiar dintr-o aplicație care nu este scrisă în Java și rulează pe o altă platformă.

În ciuda compatibilității naturale între CORBA, XML și Java, poate ați observat că analizoarele XML nu sunt servere CORBA care să ofere servicii de analiză cerute de un client CORBA. De fapt, sunt obiecte Java obișnuite. Se întâmplă așa deoarece există unele cheltuieli asociate cu trecerea prin CORBA, mai ales dacă transferați șiruri lungi XML înainte și înapoi. Deoarece analizoarele XML sunt ușoare și prompt disponibile, este mai logic să aveți o instanță a analizorului disponibilă pe calculatorul care are nevoie

de analiză și să conectați acel analizor direct în codul dumneavoastră, decât să emiteți o cerere la distanță.

Mai există o conexiune între XML și CORBA care nu este chiar atât de evidentă: modul în care sunt scrise specificațiile XML. Specificațiile XML oficiale, cum este specificația DOM, listează adesea nume de funcții, parametri, tipuri returnate și așa mai departe. Realizatorii de specificații ar fi putut să le listeze pentru C++, Java și așa mai departe, dar au ales o abordare mult mai simplă. Au definit interfețele utilizând notația IDL din CORBA. Deoarece s-a căzut de acord asupra unor mapări între IDL-ul CORBA și toate limbajele de programare populare, comitetul specificației trebuie să păstreze o singură listă, scrisă în IDL-ul CORBA, și nu mai multe liste diferite scrise pentru mai multe limbaje de programare.

Depanarea Java și XML

Rularea programului CGI Java

Găsirea problemelor din programul dumneavoastră CGI poate fi dificilă dacă acesta nu vrea nici măcar să pornească. Comentariul de la începutul învelișului Perl este foarte util în cazul în care programul dumneavoastră Java se comportă ciudat. Eliminați caracterul # de la începutul liniei și salvați din nou învelișul Perl, apoi rulați din nou CGI-ul dumneavoastră. Deseori, prin efectuarea acestei acțiuni puteți vedea mesajele de eroare din Java. Numai că nu trebuie să uitați să puneți la loc caracterul de comentariu după ce codul Java funcționează.

De asemenea, nu trebuie să uitați că orice este afișat la ieșirea standard de către un program CGI apare și în browser. Nu vă jenați și adăugați ieșiri de depanare prin apeluri ale metodei `System.out.println()`.

Punerea în funcțiune a unui servlet

Dacă una dintre metodele `doGet()` sau `doPost()` este deosebit de complicată, poate că doriți ca acest cod să funcționeze mai întâi ca aplicație Java în linie de comandă. Mesajele de eroare ale servlet-urilor pot fi uneori la fel ca mesajele de eroare Perl: serverul răspunde numai `500 Internal Server Error` (eroare internă în server) și vă lasă să ghiciți. După ce aplicația funcționează, copiați codul în servlet-ul dumneavoastră și continuați testarea.

Obținerea corectă a căilor de includere a claselor dumneavoastră este vitală. Pe un sistem Windows, asigurați-vă că ați setat variabila de mediu pentru calea de includere a claselor cu instrumentul System din Control Panel, că ați setat variabilele ca System, și nu User, și că ați pornit serverul Web după ce ați setat aceste variabile de mediu. Dacă serverul dumneavoastră Web este un serviciu care pornește o dată cu sistemul dumneavoastră, atunci trebuie să opriți și apoi să reporniți serverul (sau, în cel mai rău caz, să reporniți sistemul) după actualizarea căii de includere a claselor. Dacă utilizați Java Web Server, îl puteți opri prin deplasarea la URL-ul `http://localhost:9090/`, logându-vă utilizând ID-ul și parola de `admin` și executând clic pe butonul Shut Down. Apoi deschideți un prompter de sistem proaspăt (pentru a primi noile variabile de mediu) și reporniți serverul în modul descris în instrucțiunile de instalare.

Utilizarea facilităţilor server-side – ActiveX

CAPITOLUL **18**

În acest capitol

- ActiveX și COM
- Evoluția Microsoft spre standardele deschise
- Microsoft Distributed Internet Architecture 2000 (DNA) – Arhitectura Internet distribuită a Microsoft
- O pagină ASP simplă
- Transformarea în HTML pe server
- Un obiect COM care emite cod XML
- Depanarea ActiveX și XML

ActiveX și COM

ActiveX este o tehnologie construită pe baza COM, Component Object Model (modelul componentelor obiect) de la Microsoft. COM este infrastructura care permite programelor executabile (binare) să comunice între ele chiar dacă au fost scrise în limbaje diferite. DCOM, Distributed COM – COM distribuit – extinde COM astfel încât să funcționeze și între calculatoare diferite, chiar dacă rulează platforme diferite. Lucrul la COM pentru UNIX este în derulare de mult timp, dar pentru moment, COM este încă o lume a Windows-ului: Windows 95, Windows 98 și Windows NT. COM+ rulează numai pe Windows 2000. Un punct de pornire bun pentru a învăța mai multe despre COM este `http://www.microsoft.com/com/tech/com.asp`.

Controalele ActiveX pot fi înglobate într-o pagină Web la fel ca și applet-urile Java, dar sunt activate și afișate numai de Internet Explorer. Ele nu funcționează atunci când sunt afișate în Netscape. Controalele ActiveX sunt utilizate, de asemenea, și în aplicații, deseori ca parte a interfeței cu utilizatorul. Totuși, nu este necesar ca un control ActiveX să aibă o parte vizibilă – alte programe pot interacționa în continuare cu controlul și să facă în așa fel încât controlul să preia ceva din munca lor. Acest gen de control este referit uneori simplu ca „obiect COM".

Dar ActiveX poate fi și o tehnologie server-side. Microsoft (`http://www.microsoft.com/NTServer/web/exec/overview/overview.asp`) descrie tehnologia Active Server Pages (ASP – pagini server active) ca fiind „un mediu de aplicații deschis și care nu trebuie compilat, în care puteți combina HTML, script-uri și componente server ActiveX reutilizabile pentru a crea soluții de afaceri, bazate pe Web, dinamice și puternice". Aceasta este o modalitate comercială de a descrie script-uri server-side având aptitudinea de a apela

componente ActiveX care să rezolve lucrurile dificile pentru sistemul dumneavoastră. ASP este suportat de Internet Information Server (IIS) și Personal Web Server (PWS), ambele de la Microsoft. Dacă nu aveți acces la un server Web IIS, atunci puteți obține o copie a PWS de la `http://www.microsoft.com/Windows/ie/pws/default.htm`. Acesta este pentru Windows 95, 98 sau NT, în ciuda faptului că vine o dată cu Option Pack-ul NT 4.0.
Interacțiunea cu un control ActiveX sau cu un obiect COM dintr-o pagină ASP este un lucru simplu, așa cum veți vedea în acest capitol. Dar de unde obțineți controalele? Puteți scrie propriile dumneavoastră controale, dar un număr mare de obiecte COM sunt furnizate cu Windows sau alte produse Microsoft. Un obiect COM care este utilizat în exemplele din acest capitol este analizorul XML de la Microsoft, MSXML. Acesta face parte din Internet Explorer 5. Înainte de a continua, asigurați-vă că IE 5 este instalat pe serverul Web. Dacă nu îl aveți, atunci îl puteți obține de la `http://www.microsoft.com/windows/ie/`. În ciuda faptului că poate părea ciudat să instalați un browser Web pe un server, este cea mai simplă metodă de a instala obiectul COM analizor XML.
Merită menționat faptul că deciziile de proiectare inițiale din spatele MSXML au fost luate presupunând că acesta va fi utilizat pe partea clientului, și nu pe partea serverului. Versiunile viitoare de MSXML vor rezolva problemele de performanță și scalabilitate server-side.

Versiuni încă nelansate ale analizorului XML de la Microsoft

Microsoft a fost rapid în includerea lui XML în browserul său Web, Internet Explorer. Din păcate, unele părți importante ale specificațiilor, cele mai notabile fiind XSL și XSLT, s-au modificat de la data apariției lui IE 5. Rezultatul este că suportul XML din IE 5 nu este în totalitate conform cu specificațiile și recomandările W3C.
În ciuda faptului că IE 5 a apărut deja și că nu se va modifica în mod semnificativ, puteți obține versiuni mai recente ale analizorului XML Microsoft, MSXML 2, dar fiind conștient că lucrați cu un cod încă nelansat. Sistemele de producție de pe serverul dumneavoastră Web public nu ar trebui să utilizeze coduri nelansate. Puteți obține o copie astfel încât să fiți obișnuit cu utilizarea produsului atunci când va fi lansată o versiune stabilă a sa. În unele situații, este posibil să acceptați riscul de a lucra cu material neterminat, deoarece aveți nevoie de o versiune de MSXML care să suporte o anumită caracteristică. Este alegerea dumneavoastră.
Versiuni beta ale MSXML puteți obține de la `http://msdn.microsoft.com/downloads/webtechnology/xml/msxml.asp`.

Evoluția Microsoft spre standardele deschise

Microsoft nu a fost văzută întotdeauna ca militând pentru standardele deschise dezvoltate de altcineva. Multe dezvoltări Microsoft sunt standarde *de facto* numai datorită faptului că au fost adoptate de un număr mare de utilizatori. Nu este același lucru cu dezvoltarea deliberată de către Microsoft a unui produs în concordanță cu standardul altcuiva.
Dacă vă uitați înapoi la istoria browserelor și standardelor Microsoft, nu este o priveliște prea plăcută. Fie că erau diversele standarde HTML, foile de stil în cascadă, JavaScript versus JScript sau VBScript, Dynamic HTML sau primele schițe XML, Internet Explorer avea tendința de a realiza lucrurile mai degrabă în maniera Microsoft decât în maniera standard. Situri Web precum `http://www.webstandards.org/` conțin niște detalii crude. Dar așa cum se spune, ce a fost a fost.

Microsoft a făcut un angajament public semnificativ pentru procesul de standardizare. Acesta presupune atât ajutorul pentru crearea de standarde, cât și scrierea de software care să corespundă standardelor, chiar dacă acestea nu corespund complet intențiilor inițiale. Deși se poate încă reproșa că această concordanță este mai mică decât ar trebui, faptul că există deja un anumit nivel de concordanță este o îmbunătățire serioasă în viziunea multor observatori.
Istoria suportului XML în IE 5 este o bună ilustrare a modului în care Microsoft își adaptează software-ul la standarde, chiar și la cele care se modifică după lansarea software-ului.

Microsoft Distributed Internet Architecture 2000 (DNA) – Arhitectura Internet distribuită a Microsoft

Windows DNA este o arhitectură și o viziune construită în jurul lui Windows 2000 pentru ceea ce Microsoft numește Business Internet. Seria de instrumente de dezvoltare Microsoft – Visual Studio – asigură crearea unui suport DNA complet. COM+, succesorul lui COM, este modul în care aplicațiile și componentele comunică între ele și își împart munca. Deciziile de proiectare din spatele DNA se bazează pe realitatea software-ului pentru rețea: uneori conexiunile sunt lente, alteori celălalt calculator este indisponibil, iar performanța este întotdeauna supusă riscurilor în cazul în care aplicațiile nu se scalează corect.
Despre DNA puteți citi mai multe la `http://www.microsoft.com/dna/`, dar prima întrebare evidentă este: ce are în comun cu XML? Răspunsul este: destul de multe. Iată o descriere de la Microsoft despre genurile de sisteme cărora le este destinat DNA:

Notă Afacerile se îndreaptă rapid către un model informatic standard bazat pe Web, caracterizat prin legături conectate liber între copii de sisteme concentrate pe servicii. Un procent însemnat de situri Web de afaceri – colecții de servere, aplicații și date care oferă servicii on-line – sunt create utilizând platforma Microsoft Windows DNA ca bază pentru acest model informatic.

Imaginați-vă, pentru un moment, un sistem precum cel descris de citatul de la Microsoft. Legături conectate liber – ceea ce înseamnă sisteme separate care partajează informații, dar într-un mod standard și portabil, astfel încât o legătură se poate modifica spectaculos fără a afecta codul din celelalte legături. Iar atunci când vă gândiți la partajarea de informații într-un mod standard și portabil – partajând informații care se autodescriu – este normal să vă gândiți la XML.
XML este cheia pentru a face Windows DNA să funcționeze – și nu spun acest lucru numai pentru că aceasta este o carte despre XML. Iată un alt citat Microsoft: „... poate utilizați deja Windows DNA sau unele elemente ale sale. Diferența constă în considerarea părților ca fiind un întreg integrat, legat flexibil prin XML." Microsoft a încorporat (sau va încorpora) suport XML în produsele de dezvoltare (Visual Studio), în browserul Web (Internet Explorer), în serverele de baze de date (inclusiv SQL Server 2000) și chiar în sistemul de operare (Windows 2000). Microsoft nu a cuprins nici Internetul într-un mod atât de universal și care să afecteze atât de multe produse și

echipe diferite, nici măcar la primele programe legate de Internet.

Și mai există și BizTalk Server 2000, proiectat să ușureze schimbul de documente XML într-un context de afaceri. El extinde XML, tipic pentru politica Microsoft „cuprinde-și-extinde", pentru a-l face mai util în afaceri. Despre BizTalk puteți citi mai multe în Capitolul 19, „Utilizarea brokerilor de obiecte comune".

Windows DNA 2000 este un efort mare și atotcuprinzător. Poate nu doriți să vă construiți întregul sistem în modul DNA. Totuși, este bine de știut că majoritatea, dacă nu chiar toate tehnologiile Microsoft pe care le utilizați pentru a construi sistemul dumneavoastră, au suport XML. Este cert că vă ușurează munca. Trebuie numai să procedați cu atenție pentru a vă asigura că documentele XML pe care le realizați și le citiți sunt într-adevăr portabile și corespund standardului.

O pagină ASP simplă

Active Server Pages sunt, simplu spus, metode de a rula script-uri pe server. Ele combină JavaScript sau VBScript cu HTML care va fi trimis la browser. Script-ul este executat înainte de a trimite ceva către browser, iar dacă utilizatorul folosește View Source, nu există nici un semn al codului de scriptare original.

Puterea ASP stă în obiectele pe care vi le furnizează ca să lucrați cu ele. Sunt obiecte care păstrează constante valabile în întreaga aplicație, care reprezintă o conexiune cu o bază de date sau cu unele înregistrări extrase din acea bază de date, și, poate cel mai important, care furnizează variabile de stare referitoare la sesiunea unui singur utilizator care interacționează cu serverul dumneavoastră. ASP utilizează cookie-uri pentru a spune cărei sesiuni îi aparține oricare cerere particulară.

Iată structura unei pagini ASP tipice:

```
<%@ Language=jscript%>
<%
Response.ContentType ="text/html";

if(Request("submit").Item == "Process Request")
{

// process the request and emit HTML

}
else
{

// emit HTML for a form

}%>
```


Sugestie de la LeeAnne.com Words to weave by

Codul ASP este text obișnuit, semănând cu HTML. Aceasta înseamnă că nu aveți nevoie de instrumente speciale pentru a crea pagini ASP. Deoarece utilizați un limbaj de scriptare (JavaScript, în aceste exemple), nu aveți nevoie de un instrument pentru a compila codul dumneavoastră sau a-l pregăti în alt mod pentru a-l rula. Îl scrieți, îl salvați și îl încărcați în browserul dumneavoastră Web.

Obiectul `Response` păstrează informația pe care codul dumneavoastră o trimite browserului. Acest exemplu setează numai tipul conținutului. În acest capitol veți vedea și codul care scrie codul HTML sau o altă ieșire pentru obiectul `Response`.

Obiectul `Request` păstrează anteturile cererii browserului. Puteți determina valoarea unui câmp din formular sau, ca în acest exemplu, puteți determina dacă s-a executat clic pe un anumit buton de trimitere. Paginile ASP mai sofisticate utilizează intens obiectul `Request`.

În pagina folosită ca exemplu ASP pentru acest capitol, utilizatorului i se cere să introducă un șir XML scurt care să arate astfel:

```
<name>kate</name>
```

Un nume este extras din acel cod XML și se returnează un cod XML ușor diferit. Iată codul pentru acea parte a paginii ASP care emite formularul:

```
else
{%>
        <HTML><HEAD><TITLE>Using XML</TITLE></HEAD>
        <BODY><H1>XML from a form</H1>
        Enter some XML in the form &lt;name&gt;Your Name&lt;/name&gt;:<BR>
        <FORM action=ParseString.asp method=post id=form1 name=form1>
        <input name=xmlIn> <BR>
        <input type=submit name=submit value="Process Request">
        </FORM>
        </BODY>
        </HTML>
<%}
```

Acesta este HTML curat care nu conține variabile sau calcule. ASP vă permite să combinați în acest mod HTML cu script-uri. Observați că numele câmpului `INPUT` este `xmlIn`. Atunci când utilizatorul trimite formularul, este încărcată aceeași pagină ASP. De data aceasta, testul care verifică butonul de trimitere care a fost acționat execută codul dumneavoastră de cerere de prelucrare.

Pentru prelucrarea unui formular există două părți. Una dintre ele este extragerea câmpurilor din formular, iar cealaltă este executarea unor acțiuni asupra acelor câmpuri. În pagina exemplu ASP există un singur câmp de interes. El poate fi extras astfel:

```
var xml = Request("xmlIn").Item
```

Observați că șirul transferat lui `Request()`, `xmlIn`, este numele etichetei `INPUT` din eticheta `FORM` emisă de cealaltă parte a acestei pagini.

Pentru a analiza șirul, creați un analizor pentru document, setați-l pentru operația simultană și transferați șirul metodei sale `loadXML()`:

```
var docIn = Server.CreateObject("Microsoft.XMLDOM");
docIn.async = false;
docIn.loadXML(xml);
```

Acest cod utilizează un analizor DOM – componenta MSXML suportă numai DOM. Pagina exemplu ASP se așteaptă să primească un element <name>. Iată codul care găsește acel element și primește valoarea nodului său text:

```
var nodes = docIn.getElementsByTagName("name");
if(nodes.length > 0)
{
    var submittedName = nodes.item(0).text;
}
```

Sugestie de la LeeAnne.com Words to weave by

Dacă aveți dubii în legătură cu această linie de cod DOM, poate veți dori să recitiți Capitolul 10, „DOM (Document Object Model) – Modelul document obiect".

Se așteaptă ca pagina exemplu ASP să emită un cod XML care să semene cu acesta:

```
<person>
  <name>
    <first>Kate</first>
    <last>Gregory</last>
  </name>
</person>
```

Într-o aplicație reală, codul XML emis se va baza pe valorile găsite într-o bază de date sau calculate într-un alt mod dintr-o altă intrare. În acest exemplu, codul XML simplu codat hard este suficient pentru a demonstra tehnicile. Pentru a fi mai interesant, un prenume primește un tratament special:

```
var lastName = "";
if(submittedName == "Kate"){lastName = "Gregory";}
```

Iată codul JavaScript simplu care creează codul XML pentru a fi scris la ieșire:

```
docOut = Server.CreateObject("Microsoft.XMLDOM");

var person = docOut.createElement("person");
docOut.appendChild(person);

var name = docOut.createElement("name");
person.appendChild(name);

var first = docOut.createElement("first");
name.appendChild(first);
first.appendChild(docOut.createTextNode(submittedName));

var last = docOut.createElement("last");
name.appendChild(last);
var lastName = "";
if(submittedName == "Kate"){lastName = "Gregory";}

last.appendChild(docOut.createTextNode(lastName));
```

Pentru a scrie codul HTML pentru server, o pagină ASP utilizează obiectul `Response`. În această situație, luați codul XML din obiectul document `docOut` și scrieți-l pentru răspuns:


```
//print out the contents of the tree here
var xmlString = docOut.xml;
Response.Write(xmlString);
```

Până acum, acest cod a fost foarte optimist. Ce se va întâmpla dacă analiza nu funcționează? Poate codul XML nu este valid față de DTD-ul său. Mai rău, este posibil să nu fie construit corect. Poate că este imposibil de analizat.

În unele limbaje de programare, analizorul lansează o excepție pentru a indica acest gen de probleme. JavaScript nu suportă excepțiile, deci va trebui să țineți minte să verificați codul de erori al analizorului de după instrucțiunea de analizare. Iată o abordare tipică:

```
docIn.loadXML(xml);

if (docIn.parseError.errorCode == 0)
{
    // process the XML as you wish
}
else
{
    Response.Write("<strong>XML Parsing Error</strong><BR>");
    Response.Write("File Position: " + docIn.parseError.filepos + "<BR>")
    Response.Write("Line Number: " + docIn.parseError.line + "<BR>")
    Response.Write("Column Position: " + docIn.parseError.linepos + "<BR>")
    Response.Write("Reason: " + docIn.parseError.reason + "<BR>")
    Response.Write("Error Source: " + docIn.parseError.srcText + "<BR>")
    Response.Write("File URL: " + docIn.parseError.url)
}
```

Observați utilizarea obiectului `parseError` din documentul în care a fost încărcat codul XML. În general, acestea nu au nevoie de nici o explicație: de exemplu, `line` se referă la linia în care a apărut eroarea.

Pentru a testa pagina dumneavoastră ASP, copiați-o într-un dosar care este accesibil de pe serverul dumneavoastră Web. De exemplu, eu am făcut un dosar cu numele `xml`, în `inetpub/wwwroot`. Apoi, porniți browserul Web și introduceți URL-ul paginii: URL-ul meu era `http://localhost/xml/ParseString.asp`. Modificați-l pe al dumneavoastră în consecință. După ce ați introdus URL-ul la această pagină, veți vedea formularul din Figura 18.1.

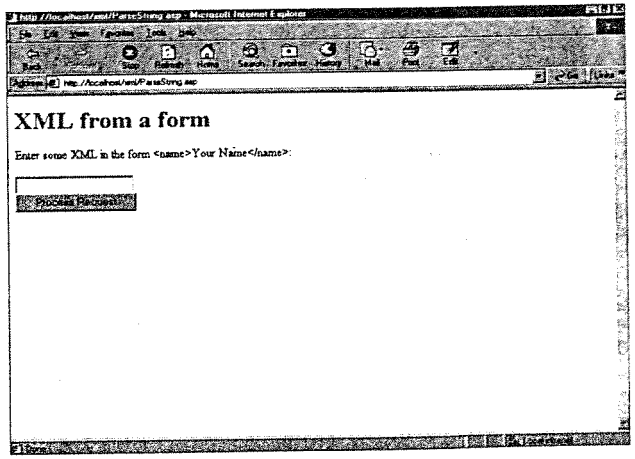


FIGURA 18.1
Pagina ASP afișează la început un formular simplu.

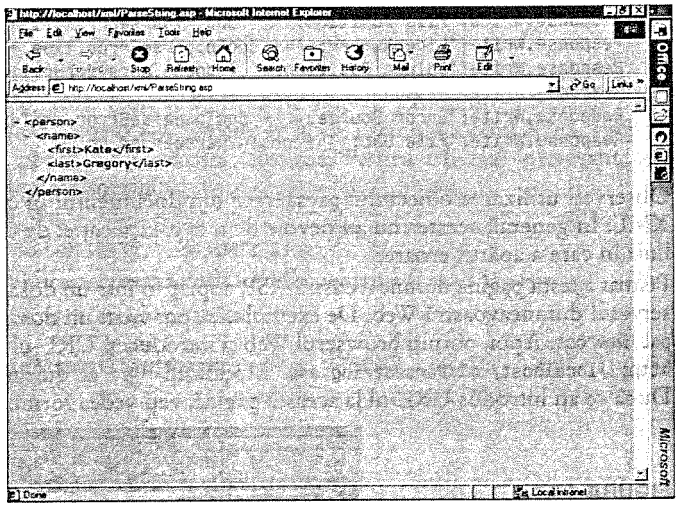
Dacă ați introdus șirul `<name>Kate</name>` în caseta de text și apoi ați acționat butonul Process Request, atunci veți vedea o ieșire XML asemănătoare cu cea din Figura 18.2. Această ieșire este cod XML, afișat ca arbore de Internet Explorer 5. IE știe că acesta este XML, deoarece tipul conținutului este setat astfel:

```
Response.ContentType = "text/xml";
```

Majoritatea fișierelor ASP setează tipul conținutului la `text/html`. Dacă procedați astfel, atunci browserul șterge toate etichetele dumneavoastră. Dacă alegeți View, Source, atunci veți vedea că etichetele sunt încă acolo, dar nu sunt afișate. Browserul le tratează ca pe un cod HTML ciudat și le ignoră. Figura 18.3 prezintă ieșirea aceleiași pagini ASP, cu tipul de conținut setat la `text/html`.

Cea de-a treia posibilitate pentru tipul conținutului este `text/plain`. Unele browsere reacționează la aceasta, în unele circumstanțe, oferindu-vă șansa de a descărca un fișier text (vezi Figura 18.4) în loc de afișarea lui în browser. Dacă alegeți să îl deschideți din acea locație, se lansează Visual Interdev (dacă este instalat), iar codul XML este deschis în acesta. Din acest motiv, este înțelept să evitați un conținut de tipul `text/plain` pentru paginile ASP.

FIGURA 18.2
Ieșirea paginii ASP este XML.



Transformarea în HTML pe Server

Scrierea unei pagini care emite cod XML este interesantă, chiar dacă nu atotcuprinzătoare. Dar codul XML afișat într-un browser nu este o priveliște plăcută. Este un mod interesant de a testa codul dumneavoastră, dacă emite sau nu codul XML pe care credeți dumneavoastră că îl emite, dar nu prea este un produs finalizat. Există trei lucruri pe care puteți să le faceți în loc să trimiteți XML brut unui browser care nu bănuiește nimic:

1. Puteți presupune că browserul este IE 5 și să adăugați o referință la o foaie de stil, astfel încât browserul să redea corect documentul XML. După cum ați învățat în Capitolul 14, „Utilizarea generației următoare de foi de stil extensibile – XSL", aceasta nu este o simplă presupunere, iar XSL sau XSLT care funcționează în IE 5 nu funcționează întotdeauna în alte contexte.
2. Puteți utiliza XSL pentru a transforma pe server codul dumneavoastră XML în cod HTML și să trimiteți codul HTML la browser. Aceasta înseamnă că indiferent de browserul folosit de utilizator, informația dumneavoastră este afișată corect.
3. Puteți scrie ceva pe partea client care cere XML și îl utilizează într-o manieră inteligentă pentru a arăta informația utilizatorului final.

FIGURA 18.3
Atunci când lui Internet Explorer i s-a spus că ieșirea este HTML, nu este afișată nici o etichetă, deși ele există în sursă.

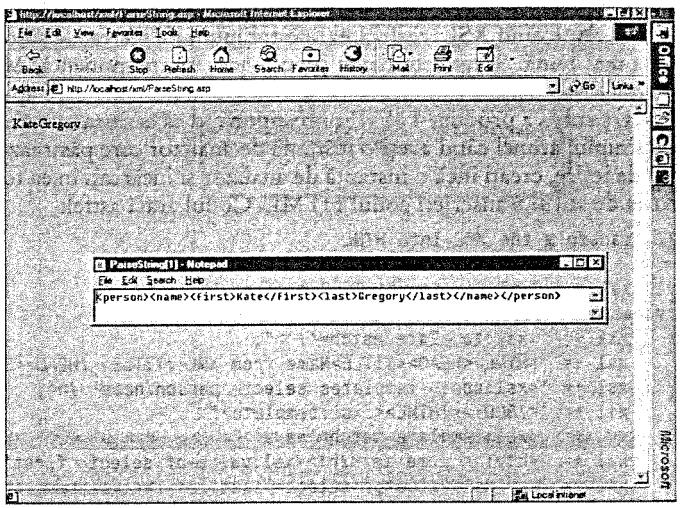
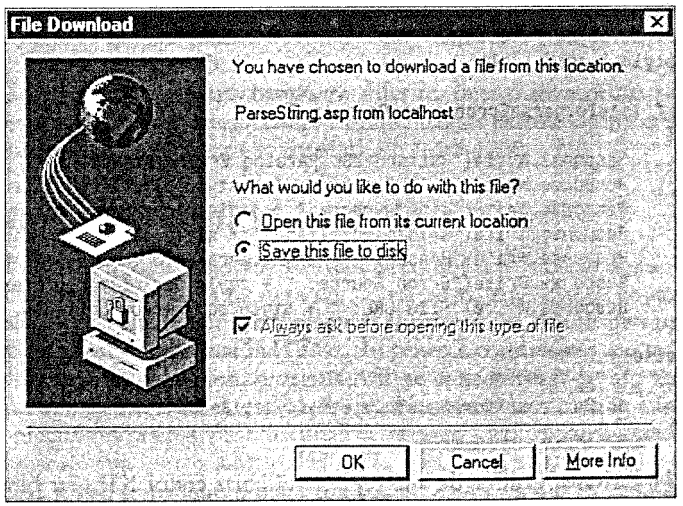


FIGURA 18.4
Atunci când lui Internet Explorer i s-a spus că ieșirea este text simplu, în loc să îl vizualizați, sunteți invitat să descărcați fișierul text.



Există câteva metode client-side pentru a obține cod XML și a-l utiliza. De exemplu, Capitolul 17, „Utilizarea facilităților server-side – Java", prezintă un applet Java care rulează pe server un servlet sau un CGI și apoi prelucrează codul XML rezultat. Nu

există nici un motiv ca același applet să nu poată primi pe server un cod XML de la o pagină ASP.

Dacă înglobați o referință la o foaie de stil în codul XML emis de dumneavoastră, atunci vă bazați pe faptul că browserul are suport XSLT. Dacă nu are, atunci utilizatorii vor vedea ceva asemănător cu Figura 18.2: un arbore XML. Probabil că nu acesta este rezultatul pe care-l doresc de la situl dumneavoastră Web sau de la aplicația dumneavoastră intranet.

În loc să vă bazați pe faptul că browserul va transforma codul dumneavoastră XML în HTML utilizând XSL, realizarea acestui lucru pe server, de către dumneavoastră, ar fi o idee mai bună. Și nu este deloc greu – aceeași componentă, MSXML, care analizează codul XML pentru dumneavoastră este și componenta care aplică foile de stil XSLT la XML pentru a produce HTML, presupunând că aceasta este ceea ce doriți. Iată cât este de simplu: atunci când aveți o instanță de analizor care păstrează arborele dumneavoastră de ieșire, creați încă o instanță de analizor și încărcați în ea foaia de stil. Aplicați apoi foaia de stil și transcrieți codul HTML. Codul arată astfel:

```
// transform the XML into HTML

var xsl = "<?xml version='1.0' ?><xsl:stylesheet
xmlns:xsl='http://www.w3.org/TR/WD-xsl'>";
    xsl += "<xsl:template match='/'>";
    xsl += "<HTML><HEAD><TITLE>Name from XML</TITLE></HEAD><BODY>";
    xsl += "<xsl:apply-templates select='person/name' />";
    xsl += "</BODY></HTML></xsl:template>"
    xsl += "<xsl:template match='name'>";
    xsl += "<h1>The name is:</h1><xsl:value-of select='first'/> ";
    xsl += "<i><xsl:value-of select='last'/></i></xsl:template></xsl:stylesheet>";

var style = Server.CreateObject("Microsoft.XMLDOM");
style.async = false;
style.loadXML(xsl);

if (style.parseError.errorCode != 0)
{
    Response.Write("<strong>XSL Parsing Error</strong><BR>");
    Response.Write("File Position: " + style.parseError.filepos + "<BR>")
    Response.Write("Line Number: " + style.parseError.line + "<BR>")
    Response.Write("Column Position: " + style.parseError.linepos + "<BR>")
    Response.Write("Reason: " + style.parseError.reason + "<BR>")
    Response.Write("Error Source: " + style.parseError.srcText + "<BR>")
    Response.Write("File URL: " + style.parseError.url)
}
else
{
    docOut.transformNodeToObject(style, Response);
}
```

Majoritatea liniilor de aici reprezintă chiar codul XSL, iar blocul de tratare a erorilor este aproape identic cu cel pe care l-ați văzut anterior în acest capitol, pentru tratarea erorilor în intrarea XML. Cele mai interesante sunt cele trei linii din mijloc care creează o altă instanță de analizor în care este încărcat codul XSL și linia de la sfârșit care transformă `docOut` din XML în conformitate cu `style` din XSLT, iar apoi îl scrie în `Response`. Totul într-o singură linie.

Figura 18.5 prezintă ieșirea acestei pagini ASP modificate. Sursa este prezentată în mica fereastră Notepad. Browserul vede numai codul HTML care a rezultat din transformare. Nu există nici o indicație că ar fi fost implicat orice cod XML sau JavaScript.

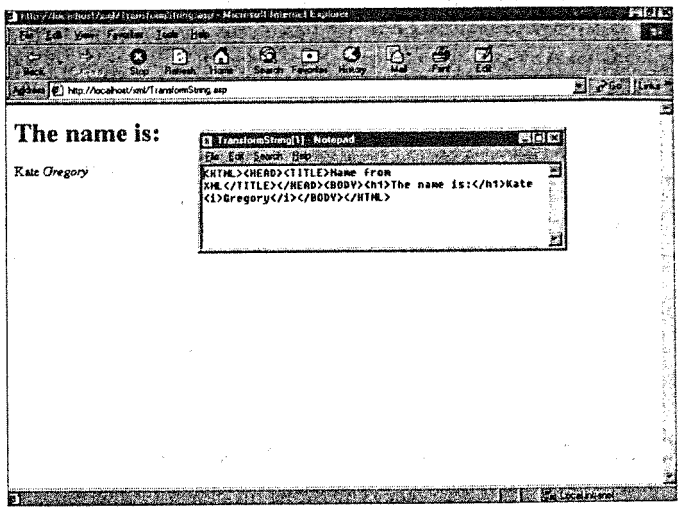


FIGURA 18.5
Atunci când pe server transformați cod XML în cod HTML, browserul nu vede niciodată codul XML original.

În ciuda faptului că acest cod este foarte simplu, serverul dumneavoastră depune mai multă muncă decât este necesar prin efectuarea transformărilor pentru fiecare caz. Altfel, puteți utiliza informația browserului în obiectul `Request` pentru a detecta browserul care este utilizat. Dacă acesta este IE 5 sau mai recent, trimiteți codul XML cu o etichetă foaie de stil și lăsați browserul să facă treaba. Dacă nu, transformați-l dumneavoastră. Ați putea chiar utiliza foi de stil diferite pentru browsere diferite, deci utilizatorii care vizualizează situl dumneavoastră pe un telefon celular primesc pagini lizibile, asemenea dezvoltatorilor care au monitoare de 21" setate la rezoluții de 1600×1200.

Un obiect COM care emite cod XML

Până acum, în acest capitol ați văzut două pagini ASP. Acestea combină logica din domeniu, cum ar fi căutarea de nume într-o bază de date sau codul XML simplificat, codat hard în exemple, cu detalii de prezentare cum ar fi macheta formularului de intrare sau metoda prin care codul XML a fost transformat în HTML. În general, combinarea acestor două lucruri nu este o idee bună. În mod ideal, o pagină ASP se concentrează pe aspect și transferă calculele, căutările în baze de date și altele la obiectele din domeniu. Un exemplu ar fi utilizarea obiectului MSXML pentru a analiza intrarea XML, pentru a crea ieșirea XML și pentru a transforma ieșirea XML în HTML. Altul ar fi să scrieți dumneavoastră un obiect COM și să îi transferați o parte din sarcini.

De ce ar trebui să utilizaţi obiecte COM din ASP

O împărţire bună a efortului implicat în această utilizare server-side a XML ar fi:

Pagina ASP cere de la utilizator cod XML.
Pagina ASP transferă codul XML unui alt obiect COM.
Obiectul COM utilizează analizorul MSXML pentru a analiza codul XML şi pentru a crea ieşirea XML (sau HTML).
Pagina ASP primeşte ieşirea de la obiectul COM şi o returnează browserului.

Separarea sarcinilor în acest mod are multe avantaje importante:

Obiectul COM este scris de obicei într-un limbaj compilat, cum ar fi C++ sau Visual Basic, care este executat mult mai repede decât JavaScript.
Obiectul COM poate fi situat pe un alt server sau clonat pe mai multe servere, reducând încărcarea pe serverul Web şi mărind scalabilitatea soluţiei.
Obiectul COM poate fi scris de cineva care cunoaşte XML şi problema din domeniu, iar aspectele vizuale (HTML) ale soluţiei pot fi scrise de cineva cu abilităţi în proiectarea de interfeţe utilizator şi în HTML.
Modificările aduse aspectului vizual al soluţiei nu necesită de obicei nici o modificare a obiectului din domeniu. La fel, modificările operaţiilor obiectului din domeniu nu necesită de obicei modificări ale codului HTML sau ASP.

Crearea unui obiect COM cu Visual Basic

Există multe limbaje în care puteţi crea un obiect COM. Visual Basic şi Visual C++ sunt cele mai populare. Lucrul în Visual Basic este puţin mai simplu pentru tipul de obiecte minore prezentate în acest capitol, dar sunteţi invitat să încercaţi realizarea unui obiect COM în Visual C++. Conceptele sunt identice cu cele prezentate aici pentru Visual Basic.

Un control ActiveX de tipul pe care noi îl numim obiect COM este, în esenţă, un DLL. Codul pe care l-aţi văzut, care creează instanţe ale obiectului analizor, este apelat într-un DLL numit MSXML.DLL. Deci pentru a construi un obiect COM în VB, începeţi prin construirea în VB a DLL-ului. Iată cum:

1. Lansaţi Visual Basic alegând Start, Programs, Microsoft Visual Studio 6.0, Microsoft Visual Basic 6.0.
2. În caseta de dialog New Project, asiguraţi-vă că este activă eticheta New, selectaţi ActiveX DLL şi executaţi clic pe Open.
3. Visual Basic creează un proiect nou, numit Project1. În fereastra de explorare Project, din partea dreapta-sus a monitorului dumneavoastră, executaţi clic-dreapta pe numele proiectului şi alegeţi Project1 Properties. Se va deschide caseta de dialog Project Properties. Aşa cum se vede în Figura 18.6, schimbaţi numele proiectului. Numele utilizat în figură este NameXML – numele utilizat de dumneavoastră pentru proiect devine numele de fişier al DLL-ului dumneavoastră, deci alegeţi un nume care are sens în contextul respectiv.
4. Visual Basic creează o clasă în proiectul dumneavoastră, numită Class1. În fereastra de explorare Project, selectaţi clasa executând clic pe numele ei. Fereastra Properties aflată chiar sub fereastra de explorare prezintă proprietăţile clasei. Selectaţi numele şi modificaţi-l în ceva mai potrivit. După cum se vede în Figura 18.7, eu am numit clasa mea ParseString.

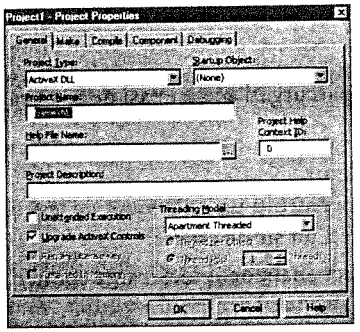


FIGURA 18.6
Modificaţi numele proiectului dumneavoastră Visual Basic din cel prestabilit, Project1.

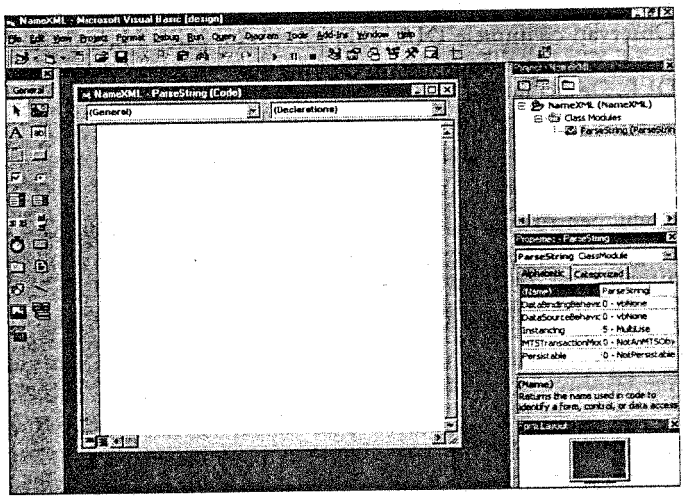


FIGURA 18.7
Modificaţi numele clasei dumneavoastră Visual Basic din cel prestabilit, Class1, într-unul mai potrivit, cum ar fi ParseString.

În acest moment aveţi un înveliş vid care poate păstra obiectul COM creat de dumneavoastră. Nu aveţi la dispoziţie nici un formular cu care să lucraţi, deoarece acest DLL ActiveX nu are o interfaţă utilizator vizuală. În schimb, adăugaţi codul clasei.

O pagină ASP rulează codul dumneavoastră prin apelul unei funcţii conţinute în DLL. Prin urmare, primul lucru care trebuie făcut este să adăugaţi la clasă o funcţie publică. În fereastra mare pentru cod introduceţi codul pentru o funcţie denumită corespunzător. Iată structura unei funcţii `ProcessName()` pentru exemplul de obiect COM:

```
Public Function ProcessName(xmlin As String) As String

        Dim returnstring As String

        If ((Len(xmlin) > 0) And Not IsNull(xmlin)) Then

        ' process the XML

        Else
```

```
        returnstring = "<error>No Input</error>"
    End If

    ProcessName = returnstring

End Function
```

Pentru a prelucra codul XML parcurgeți aceiași pași de bază ca și ASP JavaScript:

1. Creați un obiect analizor.
2. Încărcați codul XML în analizor.
3. Dacă nu există erori, atunci construiți un document de ieșire.
4. Dacă există erori, atunci scrieți un șir de eroare.

Codul pentru crearea unui obiect analizor și încărcarea codului XML în el este:

```
Dim docIn As New MSXML.DOMDocument
docIn.loadXML (xmlin)
```

Observați că numele DLL-ului, `MSXML`, este parte a instrucțiunii `New` care creează analizorul. Cea de a doua parte a acestei instrucțiuni, `DOMDocument`, este numele clasei din DLL. De aici încolo, codul este foarte asemănător cu codul JavaScript pe care l-ați văzut anterior în acest capitol.

Codul pentru verificarea condiției de eroare este:

```
If (docIn.parseError.errorCode = 0) Then
```

Codul pentru construirea unui arbore de ieșire din codul XML este:

```
Dim docOut As New MSXML.DOMDocument
Dim nodeList As IXMLDOMNodeList
Dim node As IXMLDOMNode
Dim person
Dim name
Dim first
Dim last

Set nodeList = docIn.getElementsByTagName("name")
If (nodeList.length > 0) Then
   Dim submittedName As String
   Dim lastName As String
   lastName = ""

   submittedName = nodeList.Item(0).Text

   Set person = docOut.createElement("person")
   docOut.appendChild (person)

   Set name = docOut.createElement("name")
   person.appendChild (name)

   Set first = docOut.createElement("first")
   name.appendChild (first)
   Dim sName
   Set sName = docOut.createTextNode(submittedName)
   first.appendChild (sName)
```



```
   Set last = docOut.createElement("last")
   name.appendChild (last)

   If (submittedName = "Kate") Then
      lastName = "Gregory"
   End If

   Dim lName
   Set lName = docOut.createTextNode(lastName)
   last.appendChild (lName)

   returnstring = docOut.xml
End If
```

Numele de variabile din acest cod care încep cu `I` sunt de tipul interfață COM. Acesta este modul în care analizorul MSXML este utilizat de Visual Basic. Instrucțiunile `Dim` fără tip alertează VB că un nume de variabilă este utilizat – tipul este determinat în timpul rulării. Deoarece același obiect este tratat atât ca nod cât și ca element, este mai simplu să nu depindă de un anumit tip.

În afară de tipuri (și de caracterele punct și virgulă), acest cod este identic cu codul JavaScript utilizat în pagina ASP.

Mesajul de eroare, ca și în pagina ASP, utilizează proprietatea `parseError` a documentului de intrare:

```
returnstring = "<strong>XML Parsing Error</strong><BR>"
returnstring = returnstring + "File Position: " & docIn.parseError.filepos & "<BR>"
returnstring = returnstring + "Line Number: " & docIn.parseError.Line & "<BR>"
returnstring = returnstring + "Column Position: " & docIn.parseError.linepos & "<BR>"
returnstring = returnstring + "Reason: " & docIn.parseError.reason & "<BR>"
returnstring = returnstring + "Error Source: " & docIn.parseError.srcText & "<BR>"
returnstring = returnstring + "File URL: " & docIn.parseError.url
```

După introducerea codului dumneavoastră Visual Basic, salvați proiectul. Alegeți apoi Project, References, pentru a afișa caseta de dialog References prezentată în Figura 18.8. Din listă, selectați componenta Microsoft XML – este ordonată alfabetic, așa că derulați până la sfârșitul tuturor componentelor Microsoft. Această componentă a fost adăugată în sistemul dumneavoastră atunci când ați instalat IE 5.

Sugestie de la LeeAnne.com Words to weave by

Dacă deschideți mai târziu caseta de dialog Project References, poate pentru a confirma că ați adăugat MSXML, el se află în capul listei – toate referințele utilizate de dumneavoastră sunt prezentate primele. Este posibil ca numărul versiunii să fie diferit de cel al versiunii pe care credeați că ați adăugat-o. În unele construcții ale IE 5, cele două șiruri care identifică pentru sistemul de operare acest DLL sunt diferite. Atât timp cât utilizați componenta „Microsoft XML", nu vă faceți griji cu privire la numerele de versiune.

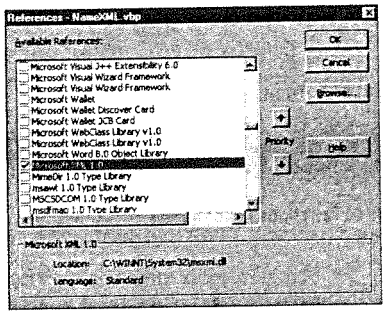


FIGURA 18.8
Adăugați analizorul XML la referințele proiectului.

În încheiere, creați fișierul DLL alegând File, Make NameXML.dll. Dacă doriți, puteți să salvați DLL-ul în dosarul proiectului dumneavoastră. În cazul în care calculatorul pe care lucrați este tocmai serverul Web pe care testați DLL-ul, ați terminat. Dacă nu, copiați DLL-ul în dosarul ales de dumneavoastră pe serverul Web, apoi înregistrați-l în sistem prin deschiderea unui prompter de comenzi în acel dosar și introducând `regsvr32NameXML.dll` (sau numele atribuit de dumneavoastră DLL-ului). Astfel înregistrați DLL-ul pe acel calculator. (VB înregistrează automat DLL-ul atunci când îl realizați.)

Sugestie de la LeeAnne.com Words to weave by

Nu încercați să utilizați DLL-ul prin rețea de pe o unitate partajată, de exemplu de pe un server central de fișiere. VB este bucuros să îl salveze acolo, dar serverul Web nu îl va utiliza din acel loc. Salvați o copie locală pe serverul Web sau urmați instrucțiunile de mai sus pentru a copia DLL-ul pe hard-discul serverului Web.

Utilizarea unui obiect COM dintr-o pagină ASP

Obiectul COM pe care l-ați creat în Visual Basic a luat logica din domeniu din interiorul paginii dumneavoastră ASP. Tot ce mai trebuie făcut este codul pentru afișarea formularului și codul pentru crearea și utilizarea obiectului COM. Iată cum arată:

```
<%@ Language=jscript%>
<%
Response.ContentType ="text/html";

if(Request("submit").Item == "Process Request")
{
        var xml = Request("xmlIn").Item;
        if((xml.length > 0) && (xml != null))
        {
                var parseObj = Server.CreateObject("NameXML.ParseString");
                Response.ContentType ="text/xml";
                Response.Write(parseObj.ProcessName(xml));
        }
        else
        {
                Response.Write("<error>No input</error>");
        }
}
else
{%>
        <HTML><HEAD><TITLE>Using XML</TITLE></HEAD>
        <BODY><H1>XML from a form</H1>

        Enter some XML in the form &lt;name&gt;Your Name&lt;/name&gt;:<BR>
        <FORM action=ParseStringWithCOM.asp method=post id=form1 name=form1>
        <input name=xmlIn> <BR>
        <input type=submit name=submit value="Process Request">
        </FORM>
        </BODY>
        </HTML>

<%}%>
```

Există trei linii care merită să fie menționate:

```
var parseObj = Server.CreateObject("NameXML.ParseString");
Response.ContentType ="text/xml";
Response.Write(parseObj.ProcessName(xml));
```

Pentru a crea un obiect COM, utilizați numele proiectului, în acest caz `NameXML`, urmat de un punct și de numele clasei din proiect, în acest caz `ParseString`. Metoda `CreateObject()` este furnizată de clasa `Server` pentru pagini ASP și este aceeași metodă utilizată și pentru crearea obiectelor analizor XML anterioare.

Tipul de conținut ar trebui setat la `text/xml` deoarece această pagină emite cod XML, și nu HTML.

Ultima linie a codului utilizează numele metodei în clasa `ParseString`, `ProcessName()`. Transferă șirul XML și scrie ieșirea direct în obiectul `Response`.

În acest moment, dacă doriți, puteți transforma codul XML în HTML utilizând o foaie de stil XSLT, așa cum ați văzut anterior în acest capitol. Această transformare nu ar trebui făcută în obiectul COM, care este preocupat numai de logica dumneavoastră din domeniu. Dacă transformați în cod HTML, asigurați-vă că modificați tipul de conținut.

Depanarea ActiveX și XML

Testarea Active Server Pages

Paginile ASP trebuie furnizate de server cu o cerere `http:`. Mulți dezvoltatori au obiceiul de a trage fișiere HTML în browser – acest lucru nu funcționează în cazul paginilor ASP. Trebuie să introduceți un URL la pagină, unul care începe cu `http:`. Pentru a vă referi la propriul dumneavoastră calculator, utilizați `localhost`.

Depanarea codului Visual Basic

Dacă ceva nu funcționează corect în codul dumneavoastră VB, pagina ASP pe care o veți scrie pe scurt afișează numai un mesaj de eroare VB. Pentru a depana codul dumneavoastră VB, urmați acești pași:

1. Opriți și porniți din nou serverul Web. Pentru aceasta puteți utiliza `restartwebserver.bat` sau alegeți Start, Settings, Control Panel și deschideți aplicația Services. Trebuie să opriți și să porniți din nou serverul Web pentru ca el să „dea drumul" DLL-ului făcut de dumneavoastră.

2. Ștergeți DLL-ul din dosarul în care l-ați salvat.
3. În VB, alegeți Run, Start. Nu par să se întâmple prea multe, dar obiectul COM rulează acum în fundal. Dacă doriți, setați un punct de întrerupere.
4. Comutați la browserul dumneavoastră Web și încărcați pagina ASP. Dacă pagina era deja încărcată, puteți executa clic pe Refresh.
5. Când execuția ajunge la punctul de întrerupere setat de dumneavoastră, aplicația activă devine VB și puteți depana aplicația dumneavoastră în modul obișnuit.
6. Când aplicația dumneavoastră VB funcționează perfect, realizați DLL-ul ca mai înainte. Nu ar trebui să porniți din nou serverul Web, dar nu veți păți nimic dacă veți proceda astfel.

Utilizarea brokerilor de obiecte comune

CAPITOLUL 19

În acest capitol

Brokeri de obiecte
Cererea serviciilor unui broker
Comunicarea între brokerii de obiecte
CORBA (Common Object Request Broker Architecture) – arhitectura brokerului de cerere de obiecte comune
COM (Component Object Model) – modelul componentelor obiect
XMLTP (XML Transfer Protocol) – protocolul de transfer XML
Java RMI (Remote Method Invocation) – apelarea metodelor la distanță
SOAP (Simple Object Access Protocol) – protocolul de acces la obiectele simple
Integrarea depozitelor de date XML – obiecte distribuite
Studii de caz

Brokeri de obiecte

Brokerii de obiecte, numiți adesea și *brokeri de cerere de obiecte* (Object Request Broker – ORB), sunt pur și simplu o modalitate de manipulare a complexității în Internet. Complexitatea aplicațiilor distribuite poate fi redusă la un nivel acceptabil, considerând orice – utilizatori, baze de date, interfețe și aplicații – ca un obiect și definind căi standard pentru a trimite mesaje între acestea. Brokerul de obiecte permite obiectelor să se găsească indiferent de locație, să discute între ele și (opțional) să translateze cererile de servicii dintr-un protocol al unui broker de obiecte în altul.

Din punctul de vedere al dezvoltatorului, integrarea ușoară a obiectelor furnizate de mai mulți producători încurajează asamblarea aplicațiilor din cele mai bune componente care sunt create și puse la dispoziție pentru un anumit scop. Astfel devin posibile tehnicile de dezvoltare rapidă a aplicațiilor, care grăbesc lansarea pe piață a produsului și permit crearea de aplicații robuste prin folosirea unor tehnologii testate deja, fără a mai fi nevoie de integrări, testări și rescrieri de cod.

De exemplu, aproape orice procesor de texte include un verificator ortografic. Majoritatea sunt create de la zero. Un dezvoltator poate utiliza eficient această resursă prin împachetarea sa ca obiect și refolosirea sa în software pentru prezentări, pentru foi de calcul sau în alte aplicații. De asemenea, ar fi un avantaj dacă ar putea fi folosite verifi-

catoarele ortografice ale mai multor producători, deoarece în acest fel ar fi posibil să creați aplicații mai mult sau mai puțin identice pentru persoane care scriu în limbi diferite, fără a fi necesar să creați un verificator ortografic pentru fiecare.

Nici o companie, oricât de mare, nu poate avea o expertiză lingvistică proprie în toate limbile pentru care ar dori să vândă procesoare de texte (adică, dacă sunt cât de cât ambițioși, pentru *toate* limbile). Utilizând o interfață obiect, orice verificator ortografic existent poate fi împachetat într-un cod astfel încât să se comporte ca un obiect, chiar dacă nu a fost creat de la început pentru un astfel de comportament.

În acest capitol vom studia modul în care se poate realiza acest lucru și modul în care obiectele distribuite peste tot în World Wide Web pot să își coordoneze activitățile și să coopereze pentru a realiza operații complexe, chiar speciale, fără a fi nevoie de crearea unui cod particular, costisitor, pentru fiecare aplicație. De fapt, veți învăța cum să încorporați mici elemente funcționale în aplicații complete, fără a scrie ceva care să semene a cod tradițional de programare. Așa cum un antreprenor poate construi o casă chiar dacă nu știe să bată un cui sau să instaleze un tablou de siguranțe, angajând oamenii potriviți, și un dezvoltator orientat spre obiecte poate construi o aplicație după un plan general, chiar dacă nu știe să rezolve sarcinile individuale cerute în acest scop.

În acest capitol vom examina brokerii de obiecte, furnizori de servicii omniprezenți în Web, și vom prezenta modul în care aplicațiile bazate pe XML pot fi integrate cu obiectele din Web. Din moment ce brokerii de obiecte depind de comunicare, iar XML este un mediu excelent de comunicare, combinarea acestora este un progres logic în evoluția standardelor majore ale brokerilor de obiecte spre XML, care este standardul preferat pentru livrarea de mesaje în Web-ul de astăzi.

Definirea interfețelor orientate spre obiecte

Bazele serviciilor oferite de un broker de obiecte depind de un *limbaj pentru definirea interfeței* (Interface Definition Language – IDL) de nivel jos, care creează interfața orientată spre obiecte pe care obiectul o prezintă lumii. O interfață descrie un set complet de operații care pot fi executate cu un obiect și mesaje care îi pot fi adresate sau pot fi așteptate ca răspuns. În termeni obiectuali, aceste operații și mesaje sunt numite *metode*.

Metodele împiedică orice alte interacțiuni cu obiectul, astfel încât operațiile care nu sunt definite de IDL sunt imposibile. În jargon obiectual, un obiect este *încapsulat* în metodele sale, astfel încât în realitate nu contează dacă obiectul are un design orientat spre obiecte. Asta înseamnă că puteți transforma în obiecte programele și bazele de date moștenite, definind un set de metode pentru fiecare. Din moment ce singura cale de acces la aplicația sau la baza de date originală, care nu este orientată spre obiecte, tratează aplicația ca și cum ar fi orientată spre obiecte, din exterior nu se observă nici o diferență.

Aplicațiile moștenite, care nu sunt orientate spre obiecte nici în interior și nici spre exterior, sunt forțate să prezinte lumii o fațadă orientată spre obiecte, prin intermediul IDL, care permite doar definiții de metode orientate spre obiecte. Un broker de obiecte poate apoi să permită altor obiecte să folosească aceste metode, ascunzând de lumea exterioară detaliile efective ale aplicației. Pot fi definite interfețe multiple pentru aceeași aplicație, ceea ce permite împachetarea funcționalității totale a aplicației în subseturi utile, care ușurează refolosirea.

Figura 19.1 vă prezintă arhitectura tipică a unei aplicații cu componente distribuite. Brokerul de obiecte este liantul care ține toate elementele la un loc.

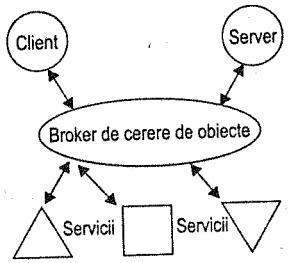


FIGURA 19.1
Aici este prezentat modelul de bază pentru serviciile care utilizează obiecte distribuite.

În Figura 19.1, învelișul IDL nu se vede, deoarece se presupune că acesta este înfășurat în jurul pictogramei serviciului, ascunzând complet procesele și reprezentările interne utilizate pentru a realiza acest serviciu și prezentând o față uniformă pentru lumea exterioară. Desigur, obiectele client și server pot fi la rândul lor definite prin IDL, brokerul tratând și aceste obiecte în mod transparent.

Norul brokerului de obiecte, care este invizibil pentru utilizator, nu este prezentat în figură. Acest broker de cerere de obiecte (ORB), care, conceptual, este o singură entitate, poate fi alcătuit de fapt din mai mulți brokeri aflați pe calculatoare diferite. Aceștia comunică între ei folosind protocoale speciale care permit unui ORB aflat la distanță să transfere cererile de servicii care nu sunt disponibile local la alte calculatoare, unde sunt disponibile.

Sugestie de la LeeAnne.com Words to weave by

Unul dintre avantajele oricărui sistem orientat spre obiecte este abilitatea de a vedea obiectele la diferite niveluri de abstractizare, astfel că perspectiva dumneavoastră asupra aplicației și asupra datelor sale poate să varieze în funcție de necesități.

Imaginați-vă ce ar putea reprezenta această diagramă generală în lumea reală. Să presupunem că obiectul client este o anumită interfață grafică cu utilizatorul (GUI), iar obiectul server este o bază de date moștenită, cu un înveliș orientat spre obiecte. Vă decideți să creați o aplicație simplă care vă va permite să interogați, să adăugați înregistrări noi și să ștergeți înregistrări vechi din baza de date. Deoarece aceste aptitudini sunt disponibile prin învelișul orientat spre obiecte al bazei de date, trebuie să găsiți (sau să scrieți) trei module pentru:

- Citirea unei înregistrări din baza de date.
- Adăugarea unei noi înregistrări în baza de date.
- Ștergerea unei înregistrări din baza de date.

Prima metodă va fi disponibilă pentru oricine, iar ultimele două vor fi disponibile doar pentru administratorii de sistem. Nu trebuie să cunoașteți exact cum arată datele din baza de date moștenită, și nici măcar să le știți locația, pentru a efectua aceste operații, atâta timp cât aveți un mijloc eficient de comunicare cu un serviciu care dispune de, sau poate să găsească, aceste cunoștințe.

Prin urmare, aplicația are nevoie doar de liantul dintre aceste părți componente. Aveți nevoie de un anumit mesaj între GUI și brokerul de obiecte care cere un anumit serviciu legat de o anumită bază de date și de un mesaj corespondent de la brokerul de obiecte care returnează ceea ce trebuie afișat ca rezultat al operației. Modalitatea exactă în care

se efectuează operațiile nu vă privește, iar brokerul nici nu v-o explică. Aceste operații pot exista în codul asociat bazei de date sau se pot afla în locații situate la mare distanță și să fie aplicate bazei de date din exterior.

Nici structura exactă a bazei de date nu este relevantă pentru problema pe care o discutăm. Poate fi cu fișier plat, relațională, orientată spre obiecte, binară, cu funcții hash, vector rar sau orice altă variantă. Mediul de stocare poate fi disc, bandă magnetică, cub optic, cartele perforate sau tăblițe de piatră. Pentru broker este indiferent, deoarece acesta are nevoie doar de capacitatea de a realiza câteva operații prin metode care sunt definite uniform în toate aceste tipuri de baze de date.

Ca regulă generală, o interfață ar trebui să realizeze o singură sarcină, sau câteva sarcini înrudite. Prin urmare, dacă o anumită bază de date este accesată în cele din urmă printr-o interfață proprie care permite sute de operații, interfețele orientate spre obiecte care îi sunt asociate trebuie să extragă subseturi mici din funcționalitatea totală. Încercarea de a împacheta prea multe funcționalități într-un obiect îi limitează aplicabilitatea la probleme generale, iar obiectul aproape sigur nu va fi refolosit. Prin separarea funcțiilor, obiectele vor fi utilizate în viitor mai prompt, iar depanarea va fi mai ușoară prin testarea funcționalităților individuale. Deși subiectul analizei sistemelor orientate spre obiecte nu este tratat în această carte, gândiți-vă să împărțiți o problemă de dimensiuni mari în mici sarcini logice pe care să le puteți atribui unor persoane care au puțin timp liber și nu veți greși prea mult.

Referințe bune legate de acest subiect complex includ:

- *Application (Re)Engineering: Building Web-Based Applications and Dealing With Legacies and Object-Oriented Client/Server Internet Environments* de Amjad Umar
- *Building Application Frameworks: Object-Oriented Foundations of Framework Design* de Mohamed Fayad și Douglas C. Schmidt
- *Design Patterns: Elements of Reusable Object-Oriented Software* de Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides și Grady Booch
- *COM and CORBA Side by Side: Architectures, Strategies and Implementations* de Jason Pritchard

Cererea serviciilor unui broker

Clienții inițiază cererile de servicii de la un broker de obiecte împachetând cererea într-un anumit mod și trimițând-o brokerului. Două dintre cele mai cunoscute tehnologii broker, COM și CORBA, au un caracter binar, deci cererile acestora nu sunt transparente și nu pot fi înțelese fără a cunoaște în amănunt interfețele. Depanarea unei interfețe defecte necesită procurarea sau dezvoltarea unor instrumente specializate. De asemenea, suportul pentru ambele platforme COM și CORBA întâmpină dificultăți de translatare. Desfășurarea rapidă a operațiilor, testarea și rescrierea, sunt împiedicate de unele probleme legate de reprezentarea internă a numerelor, de incompatibilități metodologice și de alte obstacole. Aceste două tehnologii, CORBA și COM, vor fi discutate în secțiunile „CORBA (Common Object Request Broker Architecture) – arhitectura brokerului de cerere de obiecte comune“ și „COM (Component Object Model) – modelul componentelor obiect“ din acest capitol.

Protocolul de transfer XML (XMLTP), un nou standard de broker care apare din lumea XML, adaugă virtuțile transparenței depline și aderenței la un DTD XML, astfel încât sintaxa cererilor poate fi verificată înaintea executării acestora. În plus, XMLTP folosește un mecanism de transport HTTP pentru a trece de protecții firewall și de alte bariere ale comunicațiilor în rețea. Dezvoltatorul este ajutat și de disponibilitatea schemelor standard XML și a instrumentelor de transformare care permit suportul pentru mai multe tehnologii. Deși XMLTP este încă într-un stadiu incipient, această tehnologie nouă facilitează concilierea XML și a lumilor orientate spre obiecte cu UML și cu alte instrumente de dezvoltare rapidă a aplicațiilor. Acest standard este examinat în secțiunea „XMLTP (XML Transfer Protocol) – protocolul de transfer XML“ din acest capitol, alături de câteva seturi de standarde similare: SOAP, WIDL plus XML și XIOP.

În final, tehnologia Java pentru brokerii de obiecte, RMI, care este de asemenea un standard binar, dar de o categorie foarte ușoară, deoarece presupune un mediu Java la ambele capete, este discutată în secțiunea „Java RMI (Remote Method Invocation) – apelarea metodelor la distanță“.

Comunicarea între brokerii de obiecte

O problemă a tehnologiei brokerilor de obiecte este faptul că brokeri diferiți, în cazul în care reușesc să comunice, nu comunică foarte ușor. Astfel a apărut o ocazie pentru producătorii de software de translatare dintr-un format al unui broker de obiecte în altul, în special CORBA și COM, deoarece acestea sunt mediile cele mai obișnuite pentru brokeri.

IONA, Visual Edge și Noblenet au lansat seturi de implementări ale comunicațiilor COM/CORBA care permit dezvoltatorilor să asigure suport pentru mediile mixte. Utilizarea mediilor mixte pare a fi soluția cea mai rezonabilă a dilemei în care sunt puși dezvoltatorii și marile companii: alegerea standardului. Cea mai bună soluție poate fi alegerea ambelor medii. De fapt, SOAP, XMLTP și alte standarde de transport de mesaje oferă de asemenea soluții pentru comunicarea dintre brokeri.

Din moment ce atât COM cât și CORBA folosesc protocoale de comunicare binare, conversia directă între cele două tipuri de mesaje ale brokerilor este dificilă, iar depanarea conversiilor cere o grijă deosebită. Translatarea mesajelor într-un format intermediar, în special într-un standard bazat pe XML care este transparent pentru verificări, facilitează succesul conversiei.

Având în vedere omniprezența COM în calculatoarele desktop și în multe servere la nivel de întreprindere, conjugată cu disponibilitatea unor instrumente de dezvoltare excelente pentru COM, este destul de clar că acesta nu va pierde complet segmentul de piață în favoarea CORBA.

În același timp, CORBA este un produs mult mai matur și este disponibil pentru atât de multe platforme, încât nici COM nu va putea să își asigure o dominare a pieței. Microsoft nu a fost niciodată interesată să sprijine platformele concurente și, deși COM suportă Java Virtual Machine (JVM) ca alternativă la ActiveX, versiunea Microsoft a JVM (Microsoft VM) este, într-o anumită măsură, specifică Microsoft.

Utilizarea „extensiilor“ patentate într-o miniaplicație (applet) – sau bob (Bean) – Java alterează în mod implicit obiectul care va fi utilizat pe Internet. Trebuie luate măsuri speciale pentru a oferi mecanisme alternative care nu conțin extensii. În acest scop este

necesară testarea mediului în care se va executa miniaplicația sau bobul Java și modificarea comportamentului modulului pentru a se potrivi cu mediul. Această operație contravine ideii de mașină virtuală universală, deoarece creează mai multe versiuni pentru diferite medii. Dacă trebuie scrisă o mașină virtuală diferită pentru fiecare platformă fizică sau logică, ce rost mai are?

Pe de altă parte, dacă dezvoltați software pentru o rețea exclusiv Microsoft, atunci Microsoft Java nu este mai patentată decât ActiveX, deci o puteți utiliza în mod justificat când situația o cere. Serverele care manipulează traficul general în Internet sunt obișnuite deja să adulmece mediul browserului în care intră, deci rezolvarea unei incompatibilități în plus este o problemă obișnuită, și nu una nouă.

Să înțelegem haosul brokerilor

Cineva a spus că frumusețea standardelor este că sunt atât de multe. În mod sigur afirmația se aplică la brokerii de obiecte. Deși vom examina doar câțiva, există mulți alții. De exemplu, JavaBeans este un model de broker de obiecte proiectat special pentru mediul Java și de fapt include un alt model, Enterprise JavaBeans, care nu este chiar același lucru cu boabele (Beans) obișnuite. ActiveX de la Microsoft este un alt standard, destul de strâns legat de COM, care este la rândul său legat de Universal Plug and Play (UPnP), și care adaugă o armătură pentru transmiterea de mesaje XML la acest amestec. Java Remote Method Invocation (RMI) este un al treilea standard. Întrebarea care se pune este de ce sunt atât de multe?

Răspunsul este simplu: orice broker de obiecte introduce un înveliș de căi ocolite între obiectele care pot fi utilizate într-o anumită aplicație. Brokerii cei mai generali, cum este CORBA, sunt și cei mai ineficienți, deoarece trebuie să manipuleze multe tipuri de obiecte. În schimb, dacă, de exemplu, lucrați într-un mediu exclusiv Java, nu aveți nevoie de toată această complexitate, deoarece toate obiectele dumneavoastră folosesc aceleași mecanisme. În acest caz interfața Java RMI o ia pe scurtătură și face boabele Java să coopereze mai eficient. Într-o lume proiectată de la zero pentru a folosi un anumit standard, utilizarea interfeței native a obiectului sau a unei interfețe proiectate pentru acel mediu va economisi timp, vă va scăpa de bătăi de cap în timpul codării și depanării și, în plus, va micșora și dimensiunile aplicației.

Din nefericire, cei mai mulți dintre noi nu au plăcerea să lucreze într-un mediu complet controlabil. Puteți să vă găsiți într-o situație în care interfața unei mici baze de date client a fost proiectată pentru COM, principalul sistem de contabilitate folosește CORBA, un sistem de inventariere a pieselor folosește o interfață patentată pentru obiecte furnizată de producător, iar câteva aplicații Web folosesc JavaBeans. În lumea reală totul costă bani, timp sau ambele, și nici nu vă permiteți extravaganța de a scrie din nou sistemul de operare pentru a încerca un nou limbaj de programare.

De fapt, utilizând componente care aparțin diferitelor modele, s-ar putea să vă simplificați considerabil viața, deoarece puteți descoperi în jur obiecte gata codate care pot fi îmbinate pentru a alcătui o aplicație. Astfel, ați putea folosi controale ActiveX și mici dispozitive Visual Basic pentru a manipula afișajul Windows, și JavaBeans pentru partea Web a aplicației.

CORBA (Common Object Request Broker Architecture) – arhitectura brokerului de cerere de obiecte comune

CORBA este poate cel mai general dintre brokerii de obiecte. Acesta asigură un suport complet pentru mai multe platforme decât oricare altul, inclusiv pentru platforme Windows mai vechi pe care COM sau alte metode pentru obiecte Microsoft nu sunt disponibile. Poate vă va surprinde, mai ales pe cei familiarizați numai cu platforma Microsoft, dar CORBA nu este monolitic, ci este alcătuit din mai mulți brokeri de cerere de obiecte (ORB) ai mai multor producători, cum ar fi Inprise (VisiBroker) și IONA (Orbix 2000), pentru a numi doar doi. CORBA asigură un standard de interoperabilitate, Internet Inter-ORB Protocol (IIOP), care permite unui ORB al unui producător să interacționeze cu unul de altă proveniență prin TCP/IP, fără să se piardă din funcționalitate. IIOP este o particularizare a General Inter-ORB Protocol (GIOP), care nu este mapat pentru un anumit mediu de transport.

CORBA este controlat de un larg consorțiu de dezvoltatori și acționari din domeniul tehnologiilor orientate spre obiecte, iar arhitectura sa asigură suportul pentru metode autentic orientate spre obiecte. Spre deosebire de COM, care va fi prezentat mai târziu în acest capitol, CORBA a fost proiectat de la început pentru a fi utilizabil pe mai multe platforme și limbaje. CORBA se bazează pe o tehnologie obiect care încapsulează aplicații individuale sau obiecte care formează o aplicație vastă, într-o interfață definită printr-un limbaj comun: Interface Definition Language (IDL) – limbaj de definire a interfeței. IDL asigură mai multe modalități de apelare a serviciilor de bază ale brokerului, în funcție de nevoile aplicației ca întreg.

Mijlocul principal prin care un client apelează la serviciile unui broker este un *racord* (*stub*) IDL codat hard pentru fiecare pereche de obiecte client/server. Altfel, clientul poate crea un apel similar prin apelare dinamică, dar interfața dinamică de apelare (Dynamic Invocation Interface – DII) CORBA poate fi foarte lentă. Obiectul este accesat fie printr-un *cadru* (*skeleton*) codat hard, fie prin crearea dinamică a unui cadru similar. În ambele cazuri, nici clientul, nici obiectul server nu pot să distingă între aceste opțiuni la propriul capăt al comunicării. Dacă limbajul de programare folosit de client este orientat spre obiecte, atunci racordul codat hard poate să nu fie necesar. Analog, dacă obiectul server este inerent orientat spre obiecte, poate să nu mai fie necesar nici cadrul codat hard, deoarece limbajele de programare folosite pot apela dinamic funcționalitatea aceleiași interfețe.

Unele servicii de bază pot fi oferite de o interfață ORB accesibilă direct clienților și prin adaptoare de obiecte care conțin informațiile de bază despre obiectele server. Aceste informații pot fi folosite, de exemplu, pentru a ajuta ORB să formeze cadre dinamice cu funcționalități sporite.

Deoarece toți brokerii ORB prezintă o față identică în exterior și oferă aceleași servicii, un furnizor poate să își evidențieze produsul față de cele ale altor furnizori prin eficiența implementărilor sau prin integrarea acestuia într-un mediu de programare. Încorporarea ultimelor modificări care apar în standardul CORBA poate să-i permită să se laude, cel puțin până în momentul în care restul lumii CORBA îl va ajunge din urmă.

Utilizarea CORBA

CORBA IDL seamănă într-o mare măsură cu C++, lucru care explică de ce mulți dintre furnizorii de ORB vând și compilatoare. De asemenea, IDL face CORBA relativ inaccesibil celor care nu sunt programatori. Următorul cod reprezintă interfața IDL a unui program simplu „hello, world", atât de îndrăgit de programatori.

```
module helloWorld {
        interface simpluHelloWorld {
                string ziHello();
                };
        };
```

După cum se poate observa, seamănă mult cu codul unui program și, la fel cu codul programului, va genera o interfață binară. Orice cod care generează aceeași interfață binară este echivalent în lumea OMG IDL, iar ca rezultat, au fost dezvoltate numeroase interfețe front-end pentru CORBA care permit accesul ușor din mai multe limbaje de programare și pentru mai multe sisteme de operare. Deși este tradițională în lumea programării, această concentrare binară este contrară filozofiei XML, care cere interfețe deschise, în limbaje obișnuite, care sunt accesibile în mod direct pentru oricine dorește să le examineze. Pentru depanarea unei implementări CORBA tradiționale sunt necesare instrumente speciale și multă răbdare, deoarece codul binar trebuie descifrat pentru fiecare interfață.

Cu toate acestea, CORBA este disponibil imediat aproape oriunde și este limbajul comun în dezvoltarea distribuită pentru mediile cu sisteme de operare multiple și limbaje de programare diferite. Deoarece este proiectat astfel încât să permită încapsularea codului moștenit, CORBA oferă o modalitate curată de încorporare a unei tehnologii vechi, dar funcționale, în sisteme reproiectate.

Pentru mai multe informații despre CORBA și alte subiecte înrudite, puteți consulta:

- *Client/Server Programming with Java and CORBA, Second Edition,* de Dan Harkey și Robert Orfali
- *Java Programming with CORBA* de Andreas Vogel și Keith Duddy
- *COM and CORBA Side by Side: Architectures, Strategies and Implementations* de Jason Pritchard
- *Enterprise CORBA* de Dirk Slama, Jason Garbis și Perry Russell

COM (Component Object Model) – modelul componentelor obiect

COM (Component Object Model) este brokerul de obiecte al Microsoft, asigurând comunicațiile dintre obiecte printr-o serie de interfețe binare predefinite. DEC (Digital Equipment Corporation) numește aceeași tehnologie tot COM, dar în cazul DEC înțelesul este întotdeauna Common Object Model și probabil se justifică denumirea duală, deși acronimul COM este uneori explicat în literatura Microsoft prin termenii folosiți de DEC. Oricum, în ciuda denumirilor diferite, sunt unul și același lucru.

Notă Descrierea COM ca o veritabilă tehnologie orientată spre obiecte este controversată, deoarece nu asigură moștenirea, o proprietate de bază a orientării spre obiecte, iar aplicațiile sunt construite prin agregarea unor obiecte COM mai mici. În plus, COM nu este independent de dispozitiv și nici independent de limbaj, utilizând un conglomerat fragil și greu de înțeles de soluții ad-hoc pentru probleme particulare. Din această perspectivă, COM este doar un mecanism client-server și este desconsiderat de puriștii obiectelor.

COM este baza aplicațiilor distribuite Microsoft, dar pentru a întregi sistemele este combinat cu alte câteva tehnologii. Deoarece calculatoarele Microsoft nu sunt singure în Internet, Distributed COM (DCOM – COM distribuit) asigură servicii COM pentru UNIX, IBM MVS și alte sisteme de operare pentru calculatoare mainframe și minicalculatoare, cât și pentru câteva sisteme de operare pentru microcalculatoare non-Microsoft. În combinație cu Microsoft Transaction Server (MTS – serverul de tranzacții Microsoft), Microsoft Internet Information Server (IIS – serverul de informații Internet Microsoft), Active Server Pages (ASP – pagini server active) și Microsoft Message Queue (MSMQ – coada de mesaje Microsoft), se pot construi din componente individuale simple aplicații complexe pe platforme multiple, cu condiția ca nucleul aplicației să fie localizat pe calculatoare Microsoft. Utilizarea COM într-un mediu non-Microsoft este o încercare dificilă, deoarece cea mai mare parte a infrastructurii și suportului COM este disponibilă doar pe platforme Microsoft.

Marele avantaj al COM este că suportă în totalitate cea mai răspândită și mai folosită platformă pentru utilizatori din lume. Deși serverele Windows NT nu sunt la fel de răspândite, calculatoarele desktop client Windows sunt omniprezente. Însă acest acces instantaneu la majoritatea calculatoarelor desktop are un preț. COM nu se poate integra ușor cu limbaje de programare care nu suportă aritmetica pointerilor și, așa cum am spus mai înainte, COM nu se potrivește în mediile non-Windows din cauza lipsei de suport de la alți furnizori și a suportului lipsit de entuziasm oferit de Microsoft pentru alte arhitecturi. Solaris este singura arhitectură concurentă pentru care Microsoft a portat COM.

Microsoft a extins COM prin Distributed COM, dar această distribuție este mică. Software AG realizează un produs care implementează DCOM pentru UNIX, numit EntireX, probabil o interpretare a temeliei ActiveX din DCOM. Și alți producători oferă implementări DCOM, dar acestea sunt în majoritate produse add-on cerute de piață, iar țintele principale ale DCOM sunt serverele Windows NT la un capăt și clienții Windows la celălalt.

Acum câțiva ani, Microsoft a îmbunătățit COM cu noi caracteristici și l-a redenumit COM+, care include canale (servicii de publicare și abonare), o integrare mai bună a MTS, o echilibrare dinamică a încărcării și o coadă pentru componentele COM.

Recent, toate aceste standarde Microsoft plus BizTalk (arhitectura Microsoft de comunicare business-to-business, bazată pe XML) și alte produse Microsoft au fost încorporate sub termenul general Windows DNA. Din nefericire, Windows DNA nu este chiar complet și este mai degrabă un concept de marketing spre care se tinde decât un produs real.

Notă DNA se presupune că provine de la **D**istributed Inter**N**et Applications **A**rchitecture (arhitectura aplicațiilor Internet distribuite). Ca denumire mnemonică, este cam exagerată, având în vedere că jumătate din inițiale lipsesc, iar N-ul din centru este extras din mijlocul celui de-al doilea cuvânt. De asemenea, este o denumire puțin cam ambițioasă, deoarece sugerează că Microsoft va folosi Windows DNA ca material genetic din care se vor construi aplicațiile Internet. În ultima perioadă, explicația mnemonicii a dispărut pe neobservate din documentațiile lor și este mai mult un termen de marketing decât un produs real, deoarece nu puteți să mergeți să cumpărați un exemplar.

COM, în sine, nu asigură suportul pentru aplicații distribuite, astfel încât Microsoft a extins standardul printr-un mecanism numit Object Remote Procedure Call (ORPC – apel la distanță al unei proceduri obiect). ORPC se bazează pe Open Software Foundation Distributed Computing Environment's Remote Procedure Call (DCE/RPC – apel la distanță al unei proceduri obiect într-un mediu de calcul distribuit, al Open Software Foundation). Informații despre DCE/RPC pot fi găsite la `http://www.opengroup.org/tech/dce/`. ORPC este particularizat pentru mediul Microsoft, însă, în rest, este o implementare a standardului deschis pentru mai multe platforme. Obiectele DCOM pot avea mai multe interfețe, iar clienții DCOM accesează metodele expuse în interfață prin intermediul unui pointer în codul compilat aflat chiar la locația interfeței. Această dependență de pointeri presupune că, pentru orice modificare a clienților DCOM sau a obiectelor în general, este necesară recompilarea întregului cod.

Pentru mulți, acest tip de comportament nu poate fi numit orientat spre obiecte, însă acesta funcționează în mediul țintă al majorității aplicațiilor DCOM, care este alcătuit din platforme Microsoft omogene cu câteva interfețe bine definite spre Internet, văzut ca întreg. Oricum, conexiunile în afara rețelei sunt adesea trecute prin interfețe care elimină (sau reduc) posibilitatea interacțiunilor bidirecționale din motive de securitate, astfel încât eliminarea COM/DCOM pe parcurs nu este o mare pierdere.

Utilizarea COM

COM și DCOM sunt integrate în Windows Registry, deci necesită utilizarea identificatorilor UUID (universally unique identifier – identificator unic universal) pentru Registry, care, sincer vorbind, sunt oribili. Te poți obișnui cu ei în lumea Windows, dar, ca să nu spunem mai mult, nu sunt foarte mnemonici.

Următorul fragment de cod ar putea implementa interfața DCOM a unei aplicații distribuite simple „hello, world". Interfața este definită ca dual în linia 3, astfel încât compilatorul IDL al Microsoft să genereze atât racordul și cadrul (introduse în Registry prin UUID), cât și o intrare în biblioteca dinamică de interfețe:

```
[
  uuid(2E3BCD48-13AF-D423-167B-437BE6C131DA),
  dual
  ]
library World {
  interface helloWorld : IDispatch {
    HRESULT iaHello ([in] BSTR* instr, [out, retval] BSTR* outstr);
    }
  [
    uuid(2E3BCD49-13AF-D423-167B-437BE6C131DA)
```



```
  ]
coclass Hello {
  interface helloWorld;
  };
}
```

Deoarece în fragmentul de mai sus ați definit și o bibliotecă și o clasă, aveți posibilitatea să găsiți clasa în biblioteca asociată aplicației. Dacă nu doriți să folosiți interfața dinamică, nu mai trebuie să accesați biblioteca și vă puteți baza pe racordul și pe cadrul din Registry. În concluzie, COM suportă cu ușurință atât limbajele orientate spre obiecte, cât și pe cele procedurale care nu asigură orientarea spre obiecte.

Figura 19.2 vă prezintă o arhitectură COM tipică, în care COM este stratul de la bază, MTS și o aplicație MTS reprezintă o jumătate a aplicației, iar OLE și/sau controalele ActiveX care accesează terminalul sau o bază de date reprezintă cealaltă jumătate.

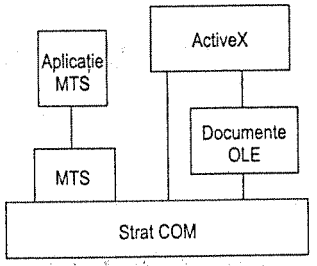


FIGURA 19.2
Aceasta este arhitectura tipică a unei aplicații bazate pe tranzacții care utilizează COM.

Deoarece COM/DCOM nu compensează și nici nu încearcă să remedieze erorile de rețea, latența sau alte probleme care pot afecta în mod negativ aplicațiile din lumea reală, este necesară o formă de eliminare a greșelilor. Procesul complet poate necesita o serie de operații care trebuie executate într-o anumită ordine și, dacă apare o eroare, întreaga serie până la punctul greșit trebuie să fie refăcută. Restaurarea după o astfel de eroare necesită înregistrarea fiecărui pas, așa încât să fie posibili pașii în sens invers care sunt necesari pentru revenirea la starea inițială. Acest proces se numește *procesarea tranzacției*.

Procesarea tranzacției

Procesarea tranzacției este destul de simplu de înțeles dacă vă gândiți la operațiile bancare. Dacă mergeți la un automat bancar să retrageți o sumă din contul dumneavoastră, trebuie să se efectueze în ordine două operații. Prima este să se retragă din contul dumneavoastră suma respectivă, iar a doua este să vă primiți banii. În orice moment este posibilă o eroare în oricare dintre aceste operații. Dacă vă primiți banii, dar aceștia nu sunt extrași din cont, banca va fi nemulțumită. Dacă banii sunt extrași din cont, dar nu îi primiți, dumneavoastră veți fi nemulțumit.

Împachetând aceste operații într-un înveliș de tranzacție, orice eroare, indiferent de momentul în care apare, blochează ambele acțiuni. De obicei, un sistem de procesare a tranzacțiilor ține evidența fiecărei acțiuni. Mai întâi, automatul va efectua un test de retragere a sumei din contul dumneavoastră. Dacă acesta reușește, automatul verifică dacă are o sumă destul de mare pentru a satisface cererea, de obicei numărând banii într-un compartiment intermediar, printr-un mecanism de siguranță. Dacă totul pare în regulă, testul de retragere a sumei va fi validat. În sfârșit, contul dumneavoastră scade cu suma retrasă, iar automatul primește semnalul să elibereze banii.

Dacă nu, tranzacția este retrasă, banii depuși în compartimentul intermediar sunt recuperați, iar automatul se închide așteptând să fie reparat. În cazul în care comunicațiile se întrerup înainte de a se recepționa semnalul de eliberare a banilor, dar după ce automatul este gata de plată, acesta ar trebui să presupună că totul este în regulă, deoarece știe că banca a aflat de tranzacție și ar putea să o reconstituie dacă este necesar. În acest moment automatul se închide deoarece nu mai poate dialoga cu banca și nu mai poate elibera în siguranță alți bani.

Aceasta este, în câteva cuvinte, procesarea tranzacțiilor, o metodă prin care se urmărește interceptarea erorilor de serializare, refacerea după astfel de erori, dacă este posibil, iar dacă nu, minimizarea pagubelor. COM/DCOM este îmbinat cu Microsoft Transaction Server (MTS) pentru a face posibilă o procesare a tranzacțiilor fiabilă în mediul Microsoft.

Pentru mai multe informații despre COM, puteți consulta următoarele texte:

- *COM and CORBA Side by Side: Architectures, Strategies and Implementations* de Jason Pritchard
- *COM Beyond Microsoft: Designing and Implementing COM Servers on Compaq Platforms* de Gene Cronin și Terence P. Sherlock
- *Web Programming with ASP and COM* de Matt J. Crouch
- *Distributed COM Application Development Using Visual Basic 6.0 and MTS* de Jim Maloney
- *COM/DCOM Unleashed* de Randy Abernethy, Jesus Chahin și Randy Charles Morin

XMLTP (XML Transfer Protocol) – protocolul de transfer XML

În mod destul de interesant, XML are în sine potențialul de a juca rolul unui broker de obiecte. Deoarece obiectele au interfețe predefinite care vă permit să le accesați, orice tip de interfață poate fi definită atâta timp cât toate obiectele convin asupra modului în care arată o interfață. Deși pare să existe o diferență între mesajele dintre procese și documentele XML, experiența UNIX, în care orice poate fi văzut ca un fișier, arată că nu este așa.

De fapt, XML are avantaje semnificative ca mediu de comunicație între procese, deoarece XML (în mod ideal) are ca referință un DTD și poate să evolueze fără să încalce versiunile anterioare. În document pot fi încorporate funcționalități suplimentare prin adăugarea unor elemente noi în DTD sau prin încorporarea altor documente XML, cu propriile lor DTD-uri, în documentul existent.

Deoarece toate documentele XML sunt analizate sintactic, nu contează cât de multă informație se adaugă, atâta timp cât informația căutată de obiectul client rămâne disponibilă. Aceasta este o diferență clară față de protocoalele de comunicații binare obișnuite, deoarece semnificația și dimensiunea fiecărui câmp sunt de obicei conținute în obiectul însuși, și nu sunt proprii mesajului. Deci, de câte ori o aplicație adaugă o funcționalitate, aplicația și toate obiectele folosite de aplicație va trebui probabil să fie recompilate și înlocuite. Într-un mediu distribuit asincron, așa cum este Internetul, șansa de a actualiza totul dintr-o dată este minimă, iar astfel de erori sunt aproape imposibil de depistat. Figura 19.3 vă prezintă diferența dintre mesajele transmise între mecanisme obiect strâns legate, cum este COM, și mecanismele slab legate care sunt posibile prin intermediul protocoalelor bazate pe XML, cum este XMLTP.

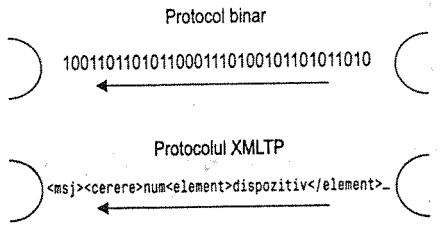


FIGURA 19.3
XMLTP promite comunicații între procese care vor fi ușor de înțeles și de depanat în cazul unor erori.

După cum puteți vedea, o interfață lizibilă pentru om este mult mai ușor de înțeles pentru programatori (sau chiar pentru muritorii de rând) atunci când creează metode între obiecte. Deoarece XMLTP se bazează pe XML, orice analizor de validare (validating parser) XML poate fi folosit pentru a verifica dacă mesajele sunt create conform DTD-ului. De asemenea, și depanarea inițială poate fi ușurată, micșorându-se astfel numărul de instrumente speciale care trebuie să fie create pentru a testa întreaga aplicație.

După cum ați văzut în discuția despre COM și CORBA, procesarea tranzacțiilor este foarte importantă în aplicațiile din lumea reală, deoarece poate să fie necesar ca un număr de operații înrudite să fie efectuate înainte ca o actualizare să poată fi considerată încheiată. Tranzacțiile în XML pot fi reprezentate simplu prin imbricare, care poate fi eventual combinată cu documente sursă externe, deoarece chiar standardul XML împiedică citirea documentelor până în momentul în care documentul însuși, adică mesajul, este complet. Prin urmare, actualizări asincrone sunt posibile și sunt automat derulate prin XML.

Pagina de bază a XMLTP se află la `http://www.xmltp.org/`, iar evoluția viitoare a acestui standard va fi anunțată aici sau pe situl dezvoltatorilor asociați de la SourceForge, la `http://sourceforge.net/project/?group_id=1893`. SourceForge oferă comunității open-source servicii gratuite de arhivare, găzduire și altele.

Multe porțiuni ale specificației nu sunt încă definite, dar deoarece întâmpină aceleași dificultăți ca și SOAP (descrise în secțiunea „SOAP (Simple Object Access Protocol) – protocolul de acces la obiectele simple"), se pare că și XMLTP va fi transportat peste HTTP pentru a permite protocolului să treacă de protecțiile firewall.

Noțiunea de firewall a fost explicată în Capitolul 16, „Securitate și confidențialitate", dar ca să recapitulăm, un firewall funcționează (parțial) prin supravegherea portului la care sosește o cerere și prin interzicerea accesului la porturi neautorizate. Administratorilor de sistem le displace în special atribuirea dinamică a porturilor, astfel încât multe protocoale dinamice eșuează inevitabil când întâlnesc un firewall, deoarece administratorii care țin la securitate nu le vor lăsa să treacă.

▶ **Vezi** „Construirea unui firewall de securitate" la **pagina 355.**

XMLTP este doar unul dintre numeroasele eforturi care se fac pentru a integra XML peste HTTP și peste funcționalitatea unui broker de obiecte. Pentru un alt exemplu, vizitați situl XIOP al lui Anders Tell, la `http://xiop.sourceforge.net/`, care este un GIOP compatibil CORBA peste serviciul de mesaje HTTP care este încă în construcție, dar este public și extrem de interesant.

Utilizarea XMLTP

Standardul XMLTP este încă într-un stadiu preliminar, deci nu prea are sens să pierdem timpul construind un exemplu imaginar. Studiați exemplul din secțiunea „Utilizarea SOAP", deoarece sintaxa va fi similară.

Java RMI (Remote Method Invocation) – apelarea metodelor la distanță

Remote Method Invocation (RMI) este o invenție a Sun Microsystems și funcționează numai într-un mediu Java. Deși Java Virtual Machine (JVM – Mașina virtuală Java), numele dat de Sun mediului de execuție Java, este proiectată pentru a fi disponibilă și unitară pe toate platformele, există probleme importante în strategia Sun. De exemplu, Microsoft a „extins" conceptul Java incluzând cârlige spre caracteristicile specifice Microsoft care sunt disponibile doar pe platformele Windows/Intel, deci ceea ce trebuia să fie un cod Java scris cu instrumente Microsoft nu mai oferă nici o garanție că va rula pe alte platforme.

Notă RMI este de fapt un mecanism de apel la distanță al procedurilor, orientat spre obiecte. Deși încorporează mult din funcționalitatea unui ORB datorită faptului că se bazează pe un limbaj care este inerent orientat spre obiecte, independența de platformă se datorează numai faptului că Java este disponibil pe majoritatea platformelor.

Spre deosebire de tehnologiile celorlalți brokeri de obiecte, RMI este chiar încorporat în JVM. Protocolul este de o categorie ușoară deoarece nu trebuie să țină cont de o mulțime de medii de execuție. RMI este un mecanism de apel la distanță al procedurilor, orientat spre obiecte, și nu un mediu complet, orientat spre obiecte, de procesare a tranzacțiilor. În orice aplicație distribuită trebuie să țineți cont de posibilitatea erorilor în rețea, de latență, concurență și de alte probleme asociate mediului nesigur al rețelei. În absența unui înveliș de tranzacție sau a unor modalități încorporate de tratare a acestor situații, utilizarea RMI presupune, pe lângă anticiparea succesului, și pregătirea unor scenarii pentru erorile posibile.

RMI alcătuiește baza pentru Jini, Enterprise JavaBeans (EJB) și JavaSpaces, fiecare dintre acestea încercând să extindă Java la un sistem complet pentru procesarea tranzacțiilor distribuite. O discuție completă despre numeroasele instrumente disponibile în universul Java Sun nu este posibilă în spațiul alocat acestui capitol, dar veți vedea un exemplu scurt de utilizare a RMI. Pentru informații suplimentare, vizitați situl Sun Java: `http://java.sun.com/`.

Utilizarea RMI

În acest exemplu, să presupunem că doriți ca un server să spună „hello" utilizând facilitățile RMI. Trebuie să definiți clasele interfață RMI ale obiectelor client și server, iar apoi să scrieți obiectele client și server.

Primul fișier Java creează o instanță a clasei interfață RMI `Remote`, operație pe care o efectuează orice aplicație RMI care dorește să folosească facilitățile RMI:

```
/* First, get the RMI base class */
import java.rmi.*;
```



```
/* Now, declare the interface and the helloRemote method */
public interface helloIF extends Remote {
  String helloRemote(String client)
         throws RemoteException;
  }
```

Următorul program Java este chiar clientul. Acesta trebuie să utilizeze interfața la distanță pe care ați declarat-o în exemplul de cod de mai sus:

```
/* First, get the base classes for RMI and RMI servers */
import java.rmi.*;
import java.rmi.server.*;
/* Second, declare the client class */
public class helloClient {
  /* Third, declare the arguments */
  public static void main(String[] arguments) {
    /* Fourth, invoke the RMI security manager */
    System.setSecurityManager(new RMISecurityManager());
    /* Fifth, call the remote server */
    try {
      helloIF x =
        (helloIF)Naming.lookup("//" + getCodeBase().getHost() +
                               "/helloServer");
      System.out.println(x.helloRemote("world."));
      }
    catch(Exception errcode) {
      System.out.println("Error: " + errcode);
      }
    System.exit(0);
    }
  }
```

În ceea ce privește serverul, acesta trebuie să utilizeze aceeași interfață cu clientul:

```
/* First, get the base classes for RMI and RMI servers */
import java.rmi.*;
import java.rmi.server.*;
/* Second, get the base classes for applets and the RMI Registry */
import sun.applet.*;
import java.rmi.registry.LocateRegistry;
/* Third, declare the server class */
public class helloServer {
  public static void main(String args[]){
    if (System.getSecurityManager() == null) {
      System.setSecurityManager(new RMISecurityManager());
      }
    try {
      helloObject remoteObject = new helloObject();
      Naming.rebind("helloObject", remoteObject);
      }
    catch(Exception errcode) {
      System.out.println("Error: " + errcode);
      }
    }
  }
```

Atât clientul cât și serverul trebuie să poată apela același obiect. Spre deosebire de un fișier Java obișnuit, acesta nu va fi compilat cu javac. În schimb, vom folosi un compilator special RMI, numit rmic, care va genera două fișiere, unul cu sufixul _stub, iar celălalt cu sufixul _skel. Programul racord (stub) este pe partea client, iar programul client îl apelează. Racordul găsește apoi serverul folosind serviciile RMI și îi transferă cererile clientului. Serverul execută obiectul cadru (skeleton), care este pe partea serverului, și transferă rezultatul înapoi racordului de pe partea client, care, la rândul său, transferă valoarea returnată clientului. Iată codul obiectului:

```
/* First, get the base classes for RMI and RMI servers */
import java.rmi.*;
import java.rmi.server.*;
/* Second, declare the object class */
public class helloObject
  extends UnicastRemoteObject
  implements helloIF {
    public helloObject() throws RemoteException {
      super();
      }
/* Now do the work */
public String helloRemote(String client)
    throws RemoteException {
      return "Hello, " + client;
      }
    }
```

Înainte de vă a putea înregistra aplicația distribuită, trebuie să porniți RMI Registry pe server. În sistemele Windows, deschideți o fereastră DOS și tastați

```
start rmiregistry
```

În sistemele UNIX, tastați

```
rmiregistry &
```

Pentru cei care nu sunt familiarizați cu UNIX, caracterul & spune interpretorului de comenzi să afișeze imediat un prompt de sistem, fără să aștepte mai întâi ca programul rmiregistry să se încheie.

În sfârșit, când rulați programele, trebuie să indicați fișierele de securitate atât pentru client, cât și pentru server. Fișierele de securitate ar trebui să arate astfel:

```
grant codeBase
"file:/someURL/" {
  permission java.net.SocketPermission
    "127.0.0.1", "accept, connect, listen, resolve";
  };
```

SOAP (Simple Object Access Protocol) – protocolul de acces la obiectele simple

Simple Object Access Protocol (SOAP) este versiunea Microsoft a unui mecanism de comunicație între procese și de apel la distanță al procedurilor (RPC), bazat pe XML. SOAP este foarte eficient în ceea ce privește trecerea prin firewall-uri. Multe RPC și protocoale de comunicații între procese nu pot trece de firewall-uri pentru a ajunge la

destinație. SOAP însă se strecoară ușor prin orice firewall care permite accesul printr-un browser Web, deoarece este atașat la HTTP, acesta trecând prin aproape orice firewall.

SOAP este parte integrantă a unei alte inițiative Microsoft legate de brokerii de obiecte XML, Universal Plug and Play (UPnP). UPnP este patronat de un consorțiu de parteneri Microsoft, dar de la anunțarea sa, în ianuarie 1999, nu prea s-a mai auzit de el, deși reprezentanțele consorțiului au prezentat produse demonstrative. Spre deosebire de CORBA și COM/DCOM, UPnP pare să fie destinat în primul rând rețelelor casnice sau unui LAN al unei mici întreprinderi și nu este un produs cu abilități generale pentru Internet. Dispozitivele din rețea se fac cunoscute emițând un mesaj la o adresă IP și la un port cunoscute, unde un proces de control colectează informații despre serviciile pe care le asigură și le înregistrează ca fiind disponibile. În termenii UPnP, acest proces este cunoscut ca publicitate și descoperire.

Notă Consorțiul UPnP are pagina de bază la http://www.upnp.org/, deși situl suferă la momentul actual de lipsa unor informații disponibile pentru public. Pare destul de logic, deoarece utilizatorii produsului sunt mai ales furnizorii de hardware, iar aceștia au acces la mai multe informații.

UPnP pare să intre în nișa ocupată de Jini, adică rețelele casnice sau comerciale de mici dimensiuni. UPnP pare proiectat pentru a intra în competiție cu tehnologia Sun bazându-se pe relațiile existente între Microsoft și producătorii și furnizorii de hardware și încercând să extindă conceptul de marketing al perifericelor Plug and Play (PnP) la dispozitivele și serviciile de rețea. Ca și Jini, UPnP nu se poate dezvolta spre Internet ca un întreg fără a întâmpina probleme.

Mecanismul de publicitate și descoperire, prin care unul sau mai multe procese de control culeg și reține informații despre toate dispozitivele care sunt disponibile în rețea bazându-se pe o serie de mesaje multicast transmise de fiecare dispozitiv în întreaga rețea, pare cam greoi pentru a putea fi extins la întregul Internet. În momentul actual nu pare să existe un mecanism pentru distribuția ierarhică a informațiilor despre dispozitive, și nici nu pare probabil ca cineva să permită unui străin să îi folosească dispozitivele fizice. În schimb, într-o rețea privată, aceste probleme dispar sau sunt mult mai mici. Numărul dispozitivelor disponibile într-o rețea casnică sau o rețea a unei întreprinderi mici este minim, iar procesul de control poate chiar să restricționeze accesul la dispozitive, astfel încât, de exemplu, programatorii să nu poată lista la imprimanta departamentului de resurse umane.

Un standard înrudit de la HP este Appliance Plug and Play (APnP), care integrează UPnP cu tehnologia fie pentru dispozitive Chai Java, cu JVM sau cu Jini de la Sun și cu orice alte limbaje pentru dispozitivele de rețea care vor apărea în viitor. Dacă Jini pare bine fixat în anumite segmente de piață, baza de furnizori care sprijină Chai APnP este neclară, dar par să fie membri ai consorțiului UPnP care vor să se asigure în caz de nereușită.

Utilizarea SOAP

Să presupunem că un program simplu pentru introducerea unor comenzi trimite o comandă pentru procurarea acestei cărți utilizând titlul, numele autorului, codul ISBN și un număr arbitrar pentru client, care poate fi folosit de aplicație pentru a indexa o bază de date ce conține clienții, în care se găsesc adresele de expediere și informații de plată.

Desigur, sunt posibile mesaje mult mai complexe, dar în acest mic exemplu nu este necesar să le prezentăm:

```
POST /LeeAnne HTTP/1.1
Host: www.leeanne.com
Content-Type: text/xml
Content-Length: whatever
SOAPMethodName: http://www.leeanne.com/usingxml/OrderEntry
<SOAP:Envelope xmlns:SOAP="urn:SOAP:schemas-soap-org:soap.v1">
  <SOAP:Body>
    <oem:OrderEntry
        xmlns:oem="http://www.leeanne.com/usingxml/OrderEntry">
      <Title>Using XML 1.0</Title>
      <Author>Lee Anne Phillips</Author>
      <ISBN>0-7897-1996-7</ISBN>
    </oem:OrderEntry>
  </SOAP:Body>
</SOAP:Envelope>
```

După completarea comenzii, poate fi returnat aproape același mesaj dar adăugând un număr de confirmare, data expedierii sau o intrare de stoc epuizat, în cazul în care, momentan, cartea nu este disponibilă.

```
HTTP/1.1 200 OK
connection: close
Content-Type: text/xml
Content-Length: whatever
SOAPMethodName: http://www.leeanne.com/usingxml/OrderConfirm
<SOAP:Envelope xmlns:SOAP="urn:SOAP:schemas-soap-org:soap.v1">
  <SOAP:Body>
    <oem:OrderConfirm
        xmlns:oem="http://www.leeanne.com/usingxml/OrderConfirm">
      <Title>Using XML 1.0</Title>
      <Author>Lee Anne Phillips</Author>
      <ISBN>0-7897-1996-7</ISBN>
      <Confirm>12345678</Confirm>
      <ShipDate>20000711T163000-8</ShipDate>
    </oem:OrderConfirm>
  </SOAP:Body>
</SOAP:Envelope>
```

Data expedierii este într-un format compatibil cu XML Schema, astfel încât să poată fi validată ușor folosind instrumente standard XML Schema. Titlul, autorul și codul ISBN sunt returnate astfel încât comanda și confirmarea să poată fi verificate manual dacă este nevoie. Figura 19.4 vă prezintă tranzacția SOAP așa cum se vede într-un browser. De fapt, tranzacția ar fi trimisă probabil direct unui alt proces.

În momentul actual, SOAP este singurul competitor real pentru un broker de cerere de obiecte bazat pe XML; totuși, câteva alternative așteaptă să își facă intrarea. XMLTP, cum am văzut mai devreme în secțiunea „XMLTP (XML Transfer Protocol) – protocolul de transfer XML", este una dintre ele, iar XIOP, discutat în aceeași secțiune, este un alt exemplu.

Combinația dintre brokerii de obiecte bazați pe XML și depozitele de date bazate pe XML va face posibilă în cele din urmă integrarea fără dificultăți și în moduri pe care acum nici nu ni le putem imagina între aplicațiile și bazele de date moștenite, pe de o parte, și instrumentele moderne orientate spre obiecte, pe de altă parte. În particular, SOAP și protocoalele de comunicație similare bazate pe XML promit o metodă de a aduce la același nivel COM/DCOM și CORBA. La baza lor, cele două protocoale sunt aproape identice și implementează aproape aceeași stivă de protocol, dar diferențele foarte mari apar la nivel de detaliu. În concluzie, translatarea între ele este posibilă.

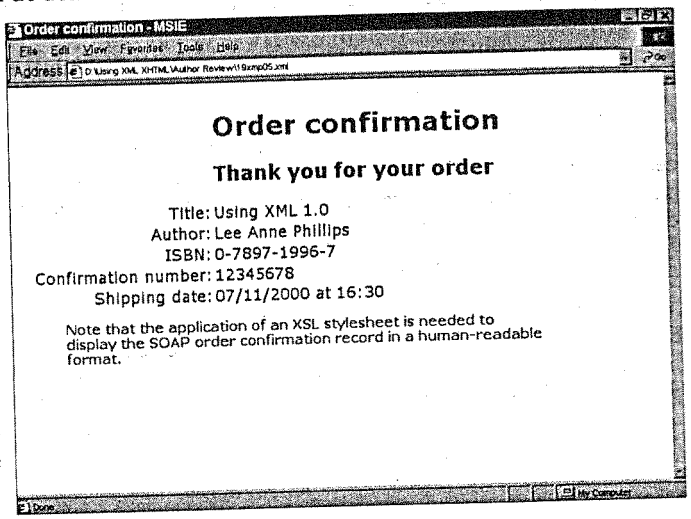


FIGURA 19.4
Această tranzacție SOAP este vizualizată în Microsoft Internet Explorer.

În ultima parte a acestui capitol vom studia modalitățile în care sistemele XML de arhivare a datelor schimbă maniera în care întreprinderile își gestionează resursele de date.

Integrarea depozitelor de date XML – obiecte distribuite

În acest capitol ați văzut câteva modalități de creare a aplicațiilor orientate spre obiecte, distribuite. Totuși, aplicațiile operează cu date în cea mai mare parte, deci în restul capitolului ne vom concentra asupra tehnologiilor pentru baze de date orientate spre obiecte (sau aproximativ) care permit aplicațiilor complet orientate spre obiecte să includă și accesul orientat spre obiecte la baze de date.

Prin utilizarea unui înveliș de obiect pentru bazele de date moștenite, se introduce un nivel de indirectare care încetinește accesul la date și poate să facă imposibilă utilizarea lor în aplicații reale. Deși poate fi o metodă valoroasă ca măsură provizorie pentru a permite dezvoltarea rapidă a unei aplicații de test reproiectate, în final va fi necesar să se convertească baza de date într-o reprezentare mai eficientă, care să permită aplicației accesul direct.

Ultima parte a capitolului examinează câteva dintre abordările cele mai inovatoare disponibile la ora actuală pentru accesul și gestionarea datelor prin XML. Sunt în curs de apariție multe alte abordări, unele dintre ele fiind ținute secrete până când proiectarea și ofertele publice ale unei firme vor fi destul de înaintate pentru a permite lansarea unui plan de marketing și de dezvoltare, înainte ca toată lumea să știe despre ce este vorba.

Pregătirea pentru un dezastru al datelor

O parte semnificativă a resurselor companiilor moderne sau ale guvernelor sunt sub formă de date, dar abilitatea noastră de a le accesa și a le utiliza este deseori, în cel mai bun caz, primitivă. Totuși, posibilitatea de a pierde aceste date, cu toate instrumentele limitate care ne stau acum la dispoziție, îi descurajează chiar și pe cei mai îndrăzneți specialiști în date ai companiilor. Datele companiilor sunt expuse la multe amenințări, de la pierderi accidentale la sabotaj.

În cazul multor tehnologii pentru copii de siguranță (backup), este imposibil să se determine dacă datele sunt valide printr-o examinare sumară. Este destul de obișnuit ca, fără știrea nimănui, o singură eroare într-un fișier de configurare să invalideze o serie completă de benzi de siguranță. Reviziile datelor sunt componente de bază ale planului, dar cea mai bună verificare este supravegherea făcută de un om. Utilizarea XML ca limbaj de siguranță permite oricărei persoane să arunce o privire peste arhivă și să își dea seama dacă datele din aceasta par rezonabile.

În cazul unei catastrofe, disponibilitatea datelor în XML, un format care permite reconstituirea datelor indiferent ce se întâmplă, chiar dacă programele și calculatoarele care să le proceseze au fost distruse, garantează posibilitatea recuperării. Acest scenariu nu este complet neverosimil. Multe aplicații rulează pe calculatoare moștenite, care acum nu se mai fabrică. Codul sursă al aplicațiilor care rulează pe acestea poate a fost pierdut sau poate nu rulează pe alte calculatoare. În schimb, XML rulează oriunde, iar instrumentele necesare pentru a-l manipula pot fi construite rapid folosind componente disponibile și standarde ca XSLT și XML Schema.

Planificarea unei stocări complet sigure

XML oferă o modalitate de stocare a datelor pentru generații la rând, garantând că datele vor fi veșnic utilizabile, iar pierderile de date sunt foarte puțin probabile. Aceasta este diferența față de multe tehnologii anterioare care aproape că au dispărut. Chiar în cazul unei simple înregistrări audio, ne-ar fi destul de greu să descifrăm un cilindru Edison, ce să mai vorbim de medii de stocare chiar mai rare. Dischetele de opt inci, destul de obișnuite acum câteva decenii, în prezent se pot găsi numai în gropile de gunoi. Deci, dacă găsiți o comoară îngropată, alcătuită din dischete gigantice conținând întreaga documentație a companiei McDonalds, toate având înscris pe etichete că au fost create pe un calculator Wang special destinat procesării de texte, aveți o problemă.

Chiar dacă utilizați un format de ieșire care se presupune că este universal, cum este Rich Text Format (RTF), nu sunteți protejat de învechire, deoarece implementările RTF ale diferiților furnizori nu sunt compatibile, iar unele nu sunt compatibile nici măcar cu încarnările lor anterioare. Formatul nu este deosebit de transparent, iar mici pierderi de date sunt posibile deoarece etichetele RTF nu se închid explicit. Și nu numai atât, după cum vedeți în următorul fragment, care cuprinde chiar începutul acestui paragraf în format RTF, sintaxa de etichetare este obscură, ca să nu spunem mai mult:

```
\par }\pard\plain \s19\ql\li0\ri0\sa240\sl240\slmult0
\widctlpar\tx7920\nooverflow\faroman\rin0\lin0\itap0
\f6\fs24\lang1033\langfe1033\cgrid\langnp1033\langfenp1033 {
Chiar daca utilizati un format de iesire care se presupune ca este universal …
\par
```

În schimb, instrumentele moderne de gestiune a resurselor de date și mediile de arhivare bazate pe XML alcătuiesc o combinație ideală pentru a maximiza controlul asupra datelor dumneavoastră și pentru a le face accesibile întregii întreprinderi sau agenții.

Gestiunea resurselor de date

Instrumentele pentru gestiunea resurselor de date permit întreprinderii să exercite un control efectiv asupra unor imperii de date de dimensiuni enorme. Chiar și utilizatorii individuali pot să aibă dificultăți în urmărirea volumului de date acumulate pe hard-discurile moderne. Dacă pe lângă hard-discurile calculatoarelor desktop mai socotiți și vectorii de discuri de scară largă din serverele de date ale companiilor, chiar și firmele de talie mică pot să dispună de teraocteți de date, împrăștiate în sute de locații.

Urmărirea datelor disponibile este dificilă chiar în situațiile cele mai bune, dar dacă aveți mai mulți angajați care trebuie să actualizeze datele, coordonarea eforturilor acestora poate fi o sarcină descurajantă dacă nu aveți un ajutor. Deci, datele pot fi nu numai pierdute sau rătăcite, ci se poate întâmpla ca două persoane să actualizeze simultan același fișier. Cine termină primul pierde, deoarece al doilea, fără să știe, va suprascrie munca primului atunci când va salva actualizările sale în fișier.

Pentru a evita aceste probleme, sistemele de control al versiunilor împiedică editarea simultană de către mai multe persoane a aceluiași fișier, dar această soluție este mult prea restrictivă în anumite situații. Secțiunea următoare, „Poet", discută o soluție pentru problema gestiunii flexibile a sistemelor complexe de documente.

Poet

Sistemele tradiționale de gestiune a resurselor, cum este Poet, sunt orientate în principal spre controlul acțiunilor membrilor individuali ai unei organizații de mari dimensiuni. Urmărind care documente sunt într-o stare stabilă și care sunt verificate de cineva, haosul posibil este înlăturat. Actualizările necontrolate sunt foarte greu de împiedicat dacă nu există un sistem formal care să oprească editarea documentului de către mai multe persoane simultan.

Poet este un sistem de gestiune a resurselor bazat pe documente XML/SGML, care permite gestiunea și urmărirea versiunilor atât pentru documente XML/SGML, cât și pentru entități binare, deși Poet nu știe nimic despre interiorul fișierelor care nu sunt XML/SGML. Poet gestionează documente la nivel de entitate, dar permite dezvoltatorului să abstractizeze documentele într-o colecție de entități logice, dacă o anumită aplicație cere acest lucru.

În majoritatea situațiilor, entitățile logice definite într-un gestionar de conținut, ca Poet, nu trebuie să reprezinte elemente dintr-un document mai mici decât cele care se editează ca o unitate. Este puțin probabil ca elementele de nivel scăzut, mai mici decât o secțiune, să formeze o unitate care poate fi editată sau refolosită. În concluzie, este perfect logic să definiți un capitol ca entitate logică într-un document carte, însă nu are sens să definiți un cuvânt scris cursiv ca entitate, deoarece este improbabil ca un singur cuvânt, sau chiar o expresie, să fie verificate în mod profitabil dacă au fost sau nu actualizate. În plus, cu cât documentul va avea o granulozitate mai fină, cu atât va fi procesat mai încet. Pe de altă parte, faptul că se permite tăierea unui document în felii aduce beneficii atunci când trebuie actualizat, deoarece acum părțile componente ale documentului pot fi editate de echipe, documentul complet fiind distribuit în siguranță între mai multe persoane care îl actualizează.

Pe lângă funcțiile de control al versiunilor încorporate în gestionarul de conținut, Poet dispune și de funcții tradiționale pentru baze de date. Puteți crea interogări pentru baze de date, care permit selectarea documentelor după criterii definite de utilizator. Astfel, este posibilă selectarea documentelor pentru procesare prin utilizarea puterii interogărilor de baze de date moderne.

Figura 19.5 vă prezintă sistemul Poet de gestiune a conținutului în timpul verificării unui document pentru actualizare. Puteți introduce un comentariu care să arate motivul actualizării și autorul acesteia, dar este posibilă și setarea aplicației astfel încât utilizatorii să deschidă sesiunea de lucru, iar identitatea lor să fie completată automat.

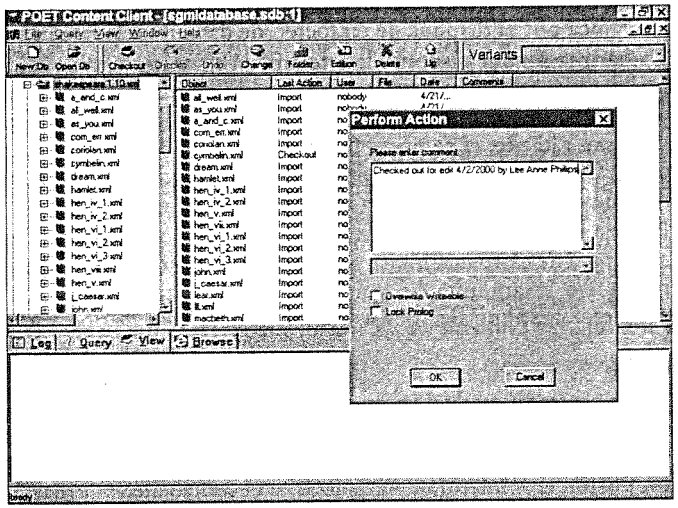


FIGURA 19.5
Gestionarul de conținut Poet prezintă verificarea unui document XML în plină desfășurare.

Poet este mai mult decât un sistem de control al versiunilor, incluzând printre altele și abilitatea de a partaja componente între mai multe documente. Această abilitate permite un tip de moștenire limitată între documentele din baza de date, deoarece componentele pot fi folosite ca bază pentru crearea altor documente. Dacă o componentă este actualizată, toate documentele care o conțin sunt actualizate automat la următoarea utilizare.

Acest tip de moștenire permite și un control centralizat asupra a ceea ce specialiștii în documente numesc „texte tipizate" (boilerplate) și folosirea acestora ca bază pentru alte documente, prin utilizarea unor componente ale lor într-un anumit document și crearea unor componente liant pentru a le integra. În multe situații, această metodă permite unei firme să răspundă rapid cererilor de informații, deoarece un document personalizat poate fi creat utilizând componente modulare care există deja.

dbXML

dbXML este o încercare de redefinire a modului în care bazele de date sunt integrate cu XML, potrivind bazele de date la limbajul XML, și nu împachetând bazele de date moștenite într-un înveliș vag înrudit cu XML. XML are în mod inerent o structură de arbore, în ciuda faptului că majoritatea bazelor de date actuale care pretind că suportă depozitele de date XML sunt fie colecții de fișiere plate, fie tabele standard indexate, cu sau fără caracteristici relaționale. Din această cauză accesul la date devine ineficient, deoarece abstractizarea unui tabel sau a unui fișier plat într-o structură de arbore necesită manipularea datelor subiacente înainte de prezentarea lor spre interfață.

Având în vedere acest lucru, dbXML are ca scop crearea unei organizări de fișier pe structura unui arbore care să oglindească structura unui fișier XML curat. În prezent, structura tipică a unui fișier de tip arbore este binară, însă documentele XML nu sunt arbori binari. Pot exista oricâte ramuri copil sub fiecare nod, deci o structură de fișier de tip arbore trebuie să reflecte ramificările multiple din XML.

Situl Web al dbXML, `http://www.dbxml.org/`, este o sursă bună pentru ultimele informații legate de dbXML și ar trebui vizitat pentru a afla dezvoltările în curs. Pentru moment, situl este puțin secretos, deoarece membrii se regrupează și decid cât anume să fie făcut public, deși se intenționează ca dbXML să fie open-source și disponibil pentru oricine. În realitatea pieței însă, dacă se publică prea multe informații prea devreme, autorul va fi dezavantajat. Un efort mai bine finanțat al unei firme comerciale poate prelua rapid un proiect în dezvoltare care nu este finanțat sau este slab finanțat. Ideile încorporate în codul open-source pot fi îngrădite de restricții de proprietate intelectuală și de patente până în punctul în care nimeni nu mai poate folosi legal conceptele originale.

Texcel

Texcel Systems din Norvegia a dezvoltat o tehnologie pentru gestiunea conținutului care încorporează conceptele de obiecte cu suport DTD și orientare spre obiecte, abstractizând structura internă a unui document independent de documentul efectiv. Este un mod de gândire puternic orientat spre obiecte și este promițător, deși nu trebuie subestimată dificultatea de a înțelege suficient de bine documentele pentru a le putea încapsula într-un mod util.

Informații foarte bune despre Texcel Information Manager (gestionarul de informații Texcel) și despre alte instrumente de gestiune a datelor pot fi găsite pe situl lor Web, la `http://www.texcel.no/`.

Ca un exemplu simplu, să presupunem că doriți să creați o bază de date cu autori dintr-o colecție vastă de documente în HL7, TEI și DocBook.

Autorii în HL7 arată astfel:

```
<PATIENTINFO>
  <NAME.GRP>
    <FIRSTNAME>Joe</FIRSTNAME>
    <LASTNAME>Blow</LASTNAME>
  </NAME.GRP>
</PATIENTINFO>
```

Pot exista mai mulți autori, deoarece HL7 a fost proiectat pentru a permite medicilor să adnoteze fișele pacienților, iar un pacient poate avea asociați mai mulți medici.

Și TEI permite existența de autori multipli, deoarece un document poate avea mai mulți autori, însă în acest caz arată astfel:

```
<titlePage>
  <byLine>by <docAuthor>Joe Blow</docAuthor></byLine>
</titlePage>
```

sau astfel:

```
<titleStmt>
     <author>Blow, Joe</author>
</titleStmt>
```

DocBook are o sintaxă puțin diferită, care arată astfel:

```
<AUTHORGROUP>
  <AUTHOR>
    <HONORIFIC>Dr.</HONORIFIC>
    <FIRSTNAME>Joe</FIRSTNAME>
    <OTHERNAME>&ldquo;The Jock&rdquo;</OTHERNAME>
    <SURNAME>Blow</SURNAME>
    <LINEAGE>Junior</LINEAGE>
  </AUTHOR>
</AUTHORGROUP>
```

Deși dumneavoastră puteți citi aceste înregistrări și vă puteți da seama ce este fiecare câmp, o aplicație trebuie ajutată pentru a-și da seama. Software-ul de gestiune a informațiilor Texcel vă permite să definiți o interfață standard orientată spre obiecte, care ascunde toate detaliile structurii interne a documentului și furnizează metode prin care puteți extrage datele în mod transparent, atâta timp cât DTD-ul documentului a fost decodat, iar informația acestuia a fost reținută.

```
interface authorInterface {
  public String Author();
}
```

Pentru utilizare, puteți defini un obiect `Doc` și o metodă de acces `getAuthor`, care returnează numele autorului indiferent care dintre cele trei DTD-uri pe care le-ați văzut descrie documentul. Apelări repetate ale metodei pot returna autorii secundari, până când un apel al metodei va returna o valoare nulă. În acel moment ați extras toți autorii.

```
public class documentInterface implements
  authorInterface {
    public String Author() {
      return Doc.getAuthor();
      }
    }
```

Cu ceva eforturi în plus, aceeași abordare poate fi folosită pentru un număr oarecare de tipuri de documente și poate fi extrapolată pentru orice date care interesează într-un domeniu de documente. Astfel, documentele dintr-un depozit se descriu singure, o îmbunătățire semnificativă față de schema descriptivă din depozitele obișnuite, în care datele descriptive sunt inserate manual de cineva. O altă situație este aceea în care depozitul depinde de numele fișierului care, pentru a fi util, ar trebui să descrie datele destul de detaliat, ceea ce este destul de greu de imaginat pentru date moștenite din sisteme de fișiere 8.3 din MS-DOS. Dacă doriți să alcătuiți o bibliografie, puteți defini o serie de metode care accesează un obiect `Doc`, ca mai jos:

```
Doc.getTitle();
Doc.getAuthor();
Doc.getPublisher();
Doc.getDate();
Doc.getISBN();
Doc.getLOC();
```

Faptul că aceste metode se comportă identic pentru mai multe tipuri de documente simplifică tratarea lor în același mod, indiferent de înfățișarea lor, la fel cum într-o bibliotecă nu contează dimensiunea cărții atunci când este înscrisă în fișă.

Aceleași tehnici orientate spre obiecte pot fi folosite pentru a ascunde faptul că datele necesare pentru o firmă se află împrăștiate în mai multe documente, de exemplu un text și adnotările sale științifice cu mai multe niveluri ale expunerii, sau chiar cu un conținut multimedia. Standardul XML nu permite legarea la documente multimedia, dar prin ascunderea documentului în spatele unei interfețe obiect, aplicația de depozitare Texcel poate să ofere o gamă mult mai largă de servicii decât XML standard:

```
Doc.getTitle();
Doc.getAuthor();
Doc.getPublisher();
Doc.getDate();
Doc.getISBN();
Doc.getLOC();
Doc.getCommentary();
Doc.getCriticism();
Doc.getAudio();
Doc.getVideo();
```

Faptul că documentul conceptual poate include o gamă largă de documente sursă efective este invizibil pentru utilizator, deoarece documentul virtual poate fi accesat uniform, ca și când ar fi un tot indivizibil.

Studii de caz

Deși discuțiile dintre partizanii COM și partizanii CORBA par fără sfârșit, adevărul este că fiecare are locul său. Într-un mediu predominant Microsoft, cu servere Microsoft NT în poziții centrale, COM, asistat de DCOM, merită luat în considerare, având în vedere existența instrumentelor de dezvoltare specializate care simplifică proiectarea și testarea aplicațiilor. Într-un mediu Linux, sau alt mediu server, în special în unul în care sunt prezente sisteme de operare și platforme diferite, CORBA este la fel de convingător. Există și situații în care are sens să fie folosite ambele, translatând la nevoie între cele două standarde.

SOAP și alte sisteme de transmitere de mesaje bazate pe XML pot comunica ușor între aplicațiile COM/DCOM – pentru care asigură multe facilități încorporate – și cele bazate pe CORBA. Din moment ce SOAP se bazează pe XML și folosește ca mecanism de transport HTTP, poate să treacă foarte ușor prin majoritatea firewall-urilor și să ajungă pe orice platformă din lume. SOAP poate folosi și FTP sau SMTP ca mediu de transport, garantând aproape total că mecanismul său de cerere și răspuns (Remote Procedure Call) poate să pătrundă într-un fel sau altul în orice sistem. SOAP poate fi folosit și pentru cereri care nu au nevoie de răspuns, printr-un protocol trimite-și-uită.

De fapt, singurul lucru care împiedică adoptarea imediată a SOAP este faptul că a fost inițial dezvoltat de Microsoft, DevelopMentor și UserLand Software. De la momentul lansării, acestora li s-au alăturat IBM și Lotus, dar suspiciunile existente în comunitatea utilizatorilor față de orice propunere Microsoft, chiar dacă este însoțită de parteneri distinși, par să aibă ca rezultat o opoziție pe mai multe niveluri față de SOAP. Microsoft se bazează pe SOAP pentru dezvoltarea cadrului BizTalk, deci este destul de evident că adoptarea generală a sistemului SOAP va fi un avantaj clar pentru Microsoft față de concurență.

Sunt propuse și alte protocoale similare în care Microsoft nu mai are avantajul terenului propriu, deci pare destul de plauzibil ca W3C să ajungă la un compromis între SOAP și XMLTP – sau alt standard concurent – înainte de a se liniști apele.

Planificarea resurselor pentru întreprinderi în XML

CAPITOLUL 20

În acest capitol

Conlucrarea XML și ERP
Comunicarea între sistemele de întreprindere
Sisteme ERP actuale
Conversii de sisteme
Depanarea sistemelor ERP

Conlucrarea XML şi ERP

Planificarea resurselor întreprinderilor (Enterprise Resource Planning – ERP) este tendința viitorului în organizațiile din întreaga lume. În mod tradițional, ERP era asociat cu seturi de aplicații care ajută o firmă să își gestioneze componentele de bază ale afacerii proprii într-o manieră integrată, fie dezvoltate cu resurse proprii, fie achiziționate ca pachet. În această definiție, ERP este limitat doar prin ceea ce consideră firma că sunt componentele de bază.

Înregistrarea comenzilor, inventarul, de asemenea fabricația, achizițiile, expedierea, serviciile client, relațiile cu furnizorii, planificarea sunt de obicei incluse, dar multe companii adaugă la acest amalgam și resursele umane și departamentul de contabilitate, într-un fel de declarație politică a importanței finanțelor și a angajaților firmei. Deoarece domeniul ERP tradițional a fost extins și include aproape orice funcție pe care firma o poate automatiza, se poate ușor include orice, iar termenul ERP devine destul de vag.

Liantul care leagă aplicațiile ERP este de obicei un sistem partajat de baze de date relaționale și un mecanism de comunicare ce permite diferitelor componente ale sistemului, cuprinzând de obicei mai multe departamente, să funcționeze ca un întreg. Deoarece XML devine rapid un standard de facto atât pentru comunicațiile din rețelele de bază cât și pentru structurarea logică a bazelor de date, XML este un instrument natural și pentru îmbinarea sistemelor ERP oferite de mai mulți producători.

Așa cum sistemele ERP actuale sunt din ce în ce mai orientate spre Web, viitoarele versiuni ERP vor include facilitățile XML de transfer al datelor ca pe un lucru de la sine înțeles. Având în vedere că afacerile devin din ce în ce mai dependente de comunicațiile business-to-business și din ce în ce mai puțin dependente de facilitățile tradiționale, „cărămidă și mortar", abilitatea de a răspunde rapid la condițiile noi și la relațiile de afaceri devine esențială pentru succesul unei întreprinderi.

Sugestie de la LeeAnne.com Words to weave by

Enterprise Resource Planning este un termen la modă care înseamnă aproape orice dorește producătorul să însemne atunci când încearcă să vă vândă un set de produse care pot să se potrivească sau nu afacerii dumneavoastră. Multe sisteme obligă firma să reechipeze și să instruiască din nou fiecare departament, ceea ce poate duce firma într-o direcție nedorită. Interesați-vă la mai mulți producători și puneți întrebări la obiect înainte de a vă decide. Puteți chiar să vă gândiți la mai mulți furnizori.

Dacă privim procesul afacerilor ca fiind continuu – integrat complet între departamente și chiar între parteneri cu afaceri deținute și gestionate separat – se creează noi ocazii și provocări pentru întreprinderile tradiționale. Îmbinarea tuturor componentelor unei întreprinderi într-un continuu uniform este dificilă, deoarece poate include:

- Planificare strategică
- Procese legate de personal și retribuire
- Contabilitate și finanțe
- Vânzări și marketing
- Creare și onorare de comenzi
- Producție și expediere
- Design și proiectare
- Relații cu subcontractorii
- Controlul inventarului
- Relații cu vânzătorii
- Satisfacerea clienților, service și asistență

Până acum, aceste activități erau compartimentate în majoritatea firmelor, diferite departamente având doar perspective parțiale asupra întregului proces. Dacă erau legături între funcții, fie ele umane sau electronice, acestea erau limitate și particularizate pentru fiecare relație în parte. Singura înțelegere completă a afacerii, în cazul în care aceasta exista, era în capetele celor din echipa superioară de management sau chiar în capul unui singur om. În cazul unor întreprinderi foarte mari, nu se poate spune că exista vreo persoană care să înțeleagă întregul proces.

Însă, în cursa spre succesul de piață, tendințele actuale spre aplatizarea structurii de management, asigurarea unor funcții în întregime din afara organizației, dependența de relațiile flexibile și în continuă schimbare cu o rețea de vânzători, contractori, organizații de marketing și vânzări distribuite, alternative de transport și așa mai departe depășesc fără speranță posibilitățile hârtiei tradiționale și chiar pe cele ale schemelor de comunicații EDI (Electronic Data Interchange – schimb electronic de date).

ERP are nevoie de o modalitate prin care aplicațiile să schimbe informații într-o manieră structurată – XML este răspunsul căutat.

Integrarea XML în ERP

Pentru a facilita atât legăturile ad-hoc dintre procese și entități, cât și legăturile permanente, este necesară o nouă concepție despre comunicațiile între procese. XML și standardele înrudite cu XML sunt mecanismele prin care pot fi eliminate comunicațiile lente cu piața și gâtuirile din traficul de date. Pornind de la sistemele de planificare a resurselor

materiale și de control al inventarelor din anii '60, ERP a crescut, incluzând o multitudine de funcții.

Chiar și acum producătorii ERP creează sisteme care consideră întreaga întreprindere ca o unitate integrată, împreună cu furnizorii, angajații și clienții săi. Comunicațiile între componentele patentate din întregul sistem se fac prin intermediul XML, deoarece facilitățile de translatare și transformare încorporate în XML îl fac să fie interfața ideală și coloana vertebrală între API și modulele de raportare ale diferitelor sisteme. De obicei, sistemele ERP includ un sistem integrat de baze de date relaționale sau alte baze de date avansate pentru a stoca datele, dar acesta nu este absolut necesar dacă sistemul asigură interfețe cu sistemele de baze de date existente.

De exemplu, Oracle, deși este recunoscut mai ales pentru sistemele sale de baze de date relaționale, este și un producător ERP. Dacă adoptați viziunea ERP a Oracle, probabil va trebui să convertiți totul în baze de date Oracle, ceea ce va afecta alte părți ale sistemelor existente, fiind necesar ca firma să înlocuiască în întregime sistemul ERP, în loc să integreze gradat sistemele moștenite sub un acoperiș comun. În plus, este posibil ca vânzătorii și contractorii să nu își poată schimba procesele lor pentru a se potrivi cu noul sistem, pierzând astfel timp cu proiectarea și implementarea unor interfețe particulare între diferitele componente ale sistemului.

SAP AG este un furnizor important de sisteme ERP SAP R/3 și, în general, este cunoscut deoarece necesită pentru instalare o reproiectare completă a proceselor existente ale afacerii, aceasta fiind o perspectivă înfricoșătoare pentru multe firme mici, care nu îl vor achiziționa, deși pe termen lung ar fi util.

Oferind mecanisme XML pentru transferul de date, producătorii pot simplifica integrarea modulelor moștenite sau recent create de alți furnizori și încorporarea lor în sisteme complete. Astfel devine posibilă atât acomodarea deplină la nevoile reale ale întreprinderii (și nu numai ceea ce poate oferi producătorul ERP), cât și migrarea într-o manieră rațională spre integrarea completă.

Limbajul comun, un vis pe cale să devină realitate

Într-o lume ideală, se presupune că toți am vorbi aceeași limbă, am avea un singur mecanism de partajare a datelor, ar exista un singur guvern, un set de legi și o organizație pentru fiecare funcție. Dar propria noastră realitate care evoluează prin diversitatea ideilor concurente, a limbilor, a culturilor și a nevoilor de afaceri este mult mai interesantă și mai robustă. Nu există evoluție fără schimbare și nu există schimbare fără diferențe.

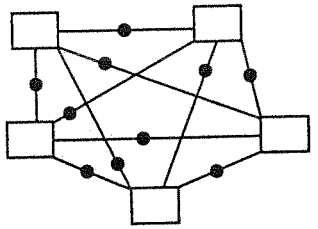
Așa cum vorbitorii de limba engleză nu vor decide să învețe toți chineza sau viceversa, nici diferitele culturi din lumea afacerilor nu vor dispărea. Ceea ce dă putere atât culturii universale cât și economiei mondiale este, în parte, imensa diversitate de puncte de vedere și de abilități care sunt cuprinse, printr-o asociere liberă, într-un sistem global.

Translatorii sunt temelia comerțului mondial și a culturii universale. Puterea XML de a transforma un vocabular în altul poate fi cel mai bine apreciată gândindu-ne la RosettaNet, cXML, eCo, Biztalk și la alte „standarde". Cu puține excepții, integrarea sistemelor diferiților producători presupune întotdeauna transformarea unui format intern în altul. Dacă ambele sisteme au o interfață XML, atunci baza unei transformări ușoare este deja prezentă. În locul unui proces îndelungat de codare particulară pentru a translata un format binar de date în altul, combinația de XSL și XSLT poate rezolva diferențe de nomenclatură fără să fie nevoie să lucrați cu limbaje de programare generale sau să atingeți un compilator.

Metoda prestabilită pentru a asigura interfața dintre două sisteme este să se construiască un program special care translatează fluxul de date dintr-un sistem în formatul nativ al celuilalt sistem. Această abordare este potrivită pentru două sau trei sisteme, sau chiar pentru o mână de sisteme, însă atunci când numărul de sisteme crește, numărul de programe de translatare se mărește fără limite. Figura 20.1 vă prezintă efectul căilor multiple de translatare. Când numărul de conexiuni crește, numărul programelor de translatare necesare crește geometric. Programele de translatare sunt reprezentate prin puncte negre. Pentru două aplicații, este necesară o singură interfață. Pentru trei aplicații sunt necesare trei interfețe. Pentru patru aplicații sunt necesare șase interfețe, pentru cinci, zece interfețe și așa mai departe.

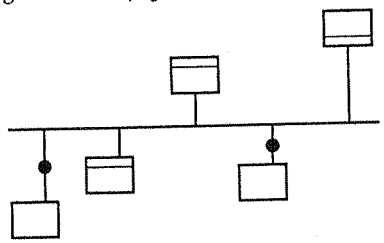
Chiar dacă la un moment dat numai două sau trei sisteme trebuie să comunice între ele, o dată cu dezvoltarea afacerii acest număr poate să crească. Curând, numărul devine imposibil de gestionat. În majoritatea cazurilor, recomandarea sau proiectarea unor sisteme care necesită interfețe individuale între mai multe sisteme este considerată o greșeală profesională. De asemenea, orice sistem care necesită astfel de interfețe, și nu le asigură doar ca opțiune legată de performanțe, este prost proiectat în termenii actuali. Acesta este unul dintre motivele pentru care toți producătorii importanți de aplicații creează sisteme care permit o interfață la un standard comun de backbone (coloană vertebrală).

FIGURA 20.1
Cinci aplicații necesită zece programe interfață.



Însă în momentul în care toate sistemele pot să fie translatate la un format comun, cum este XML, problema este mult simplificată. În locul unui program interfață pentru fiecare aplicație, este necesar numai unul pentru a translata la backbone. Dacă furnizorul asigură nativ o astfel de interfață, problema devine o bagatelă. Figura 20.2 vă prezintă arhitectura unui sistem backbone. Observați că unele dintre aplicații sunt translatate nativ la backbone (reprezentate printr-o linie subțire orizontală în interiorul aplicației), iar câteva sunt translatate printr-un program interfață particularizat (reprezentat printr-un punct negru).

FIGURA 20.2
Deoarece câteva programe sunt translatate nativ la și de la backbone, sunt necesare numai două interfețe particulare (reprezentate prin puncte negre).



Deoarece XML este proiectat pentru Web, iar XSL/XSLT este un instrument de translatare încorporat care asigură interfața dintre diferite dialecte XML, XML sau limbajele bazate pe XML constituie un backbone ideal pentru aplicațiile Web. În diagrama prezentată în Figura 20.2, aplicațiile bazate pe XML sunt reprezentate prin dreptunghiurile aplicație cu interfață nativă, deci singurele sisteme care cer interfețe particulare (puncte negre) sunt cele fără o interfață XML.

Comunicarea între sistemele de întreprindere

Deoarece majoritatea organizațiilor își dezvoltă sistemele prin incrementare, diferite departamente ale întreprinderii au achiziționat adesea – și au ajuns să se bazeze pe – sisteme de aplicații particularizate care nu comunică nativ cu restul companiei. Orice interfață între sistemele departamentale particularizate a fost realizată prin programare la comandă. Astfel, Achizițiile pot avea un sistem, Resursele umane un altul, Expedieri și recepții un altul și așa mai departe. Fiecare sistem departamental care trebuie să schimbe date cu altul este dotat cu o interfață particulară, scrisă special pentru acest scop. Deși această abordare putea să fie adecvată la momentul respectiv, imaginați-vă ce se poate întâmpla dacă un departament decide să își modernizeze sistemul.

Fiecare departament care avea o interfață particulară trebuie acum să se coordoneze cu departamentul care și-a schimbat sistemul, să rescrie fiecare program de comunicații, distribuind astfel problemele și transformând o mică modernizare într-un proiect important. Este tipul de lucrare care îi trezește cu sudori reci în miez de noapte pe informaticienii șefi și contribuie la inerția ce caracterizează multe firme.

Secțiunile următoare vă prezintă cum ar arăta un proiect de modernizare gestionat cu pricepere.

Mediul care se modernizează

Să ne imaginăm un tip oarecare de întreprindere producătoare. Aplicația inițială a întreprinderii era CA-Manman Classic (deși nu se numea astfel la momentul respectiv), de la Computer Associates. Aplicația era scrisă la început în Fortran, dar a fost particularizată pentru a se potrivi cu diferențele reale sau observate între operațiile întreprinderii și cele ale altor firme producătoare.

La un moment dat, conducerea întreprinderii a hotărât să dezvolte un sistem de control al inventarului la nivel de departament și au ales limbajul COBOL ca limbaj de dezvoltare. În momentul integrării producției cu sistemul de control al inventarului, a fost scrisă o interfață particulară în limbaj de asamblare pentru IBM AS/400, deoarece pe acesta rula sistemul de control al inventarului, iar programatorul de sisteme care răspundea de calculator în acel moment era un fanatic al limbajului de asamblare și a susținut cu insistență că, din motive de performanță, era nevoie de un limbaj de nivel jos. Între timp a părăsit compania și nimeni nu mai înțelege cum funcționează programul, deși codul fără comentarii a fost arhivat cu grijă.

Între timp, membrii departamentului de vânzări și marketing au creat un sistem de suport pentru vânzări scris în C și care rulează pe un calculator mainframe HP-3000. Când au decis să adauge o legătură cu sistemul de control al inventarului pentru ca personalul de la vânzări să poată verifica dacă un produs este disponibil, au scris o interfață într-un limbaj care rula pe HP. Nimeni nu își mai amintește care era limbajul, iar codul sursă a dispărut dintr-un motiv oarecare. Însă funcționează perfect, așa că nimeni nu vrea să îl modifice.

Când membrii departamentului de contabilitate au dorit să introducă date care sunt necesare în conturi și să trimită facturi, au fost nevoiți să introducă de la tastatură toate informațiile de la vânzări și marketing, deoarece era prea dificil să creeze o interfață cu sistemul de la marketing. Aplicațiile lor de contabilitate funcționează pe calculatoare Amdahl 5990 cu un sistem de operare compatibil MVS și au fost scrise în COBOL/CICS/VSAM.

Notă Acest exemplu poate să pară o exagerare, dar este de fapt un scenariu destul de obișnuit pentru orice organizație care există de ceva timp. Deciziile de altădată, care acum sunt aproape de neînțeles, există și vă încurcă, chiar dacă cei care le-au luat au plecat demult la alte companii.

În total, 32 de aplicații principale rulează pe șapte platforme diferite. Unele au o interfață directă cu aplicația CA-Manman iar altele nu, deși depind indirect de datele obținute din acest sistem. Această dependență poate fi prin intrări manuale sau prin legături cu programe care au legături proprii, care pot să fie la rândul lor indirecte, cu sistemul de la producție.

Și alte aplicații mici rulează probabil pe aceste platforme, dar este greu să fie identificate deoarece rulează doar ocazional, cum ar fi procesarea specială de sfârșit de an pentru inventar și contabilitate. S-ar putea ca departamentele implicate nici să nu știe unde are loc efectiv procesarea.

În acest mediu, vicepreședintele de la producție decide să migreze ramura sa din organizație spre o arhitectură distribuită, tolerantă la erori, de la Compaq, care rulează pe sisteme NonStop Integrity cu un sistem de operare Tru64 UNIX. El crede că succesorul lui CA-Manman Classic, CA-Manman/MK, poate fi făcut să ruleze în acest mediu, deci problema realizării efective a modificărilor drastice de mediu poate fi minimizată.

Strategia de migrare

Fiind impartiali și prevăzători, cei de la producție își dau seama ca această operație va avea un impact asupra multor departamente, deci cu acordul funcționarilor superiori și al celor de la finanțe și, desigur, cu aprobarea comitetului director, fiecare departament are ocazia să participe sau nu la migrarea propusă. Avantajul în acest plan este că organizația ca întreg va subvenționa parțial migrarea pentru departamentele individuale. Dezavantajul este că dacă un departament refuză migrarea, acesta va prelua costurile asociate cu menținerea platformelor care ar trebui înlocuite.

Deși sunt unele rețineri, majoritatea celor din întreprindere sunt de acord în principiu cu migrarea, mai ales având în vedere că, o dată cu accelerarea acceptării în tot mai multe departamente, costul întreținerii platformelor vechi este din ce în ce mai mare pentru cei rămași pe dinafară. Este exact ceea ce au prevăzut cei de la producție atunci când au întocmit planul.

Lucrând cu echipa Compaq eBusiness și cu Baan, producția decide să folosească o combinație de platforme Dec Alpha pe 64 de biți și Intel Xeon pe 32 de biți, pentru a profita de soluțiile ERP pentru afaceri de la Baan care există deja. Intenționează să utilizeze cadrul de lucru BizTalk de la Microsoft, bazat pe XML, pentru a asigura o mare parte a liantului ce asigură coeziunea diferitelor componente ale sistemului care nu sunt capabile nativ să comunice între ele. Deoarece Baan asigură prin produsele sale ERP componente complet echipate, cea mai mare parte din costurile asociate cu migrarea, în cazul multor departamente, este formată din costurile de achiziție și întreținere a sistemului în sine și din costurile de conversie pentru bazele de date existente.

Deși extinderea pe piață a BizTalk este încă nesigură, organizația speră că vânzătorii și subcontractorii principali vor adopta același limbaj XML și, chiar dacă aceste aprecieri pozitive ale pieței se vor dovedi incorecte, organizația crede că orice limbaj XML similar poate fi substituit cu costuri minime. Au fost concepute planuri pentru orice eventualitate, astfel încât interfețele de conversie XML să fie cât mai modularizate pentru a reduce impactul oricăror modificări.

Rezultatele inițiale ale migrării

Porțiunile Baan ale sistemului care rulează pe procesoare Intel Xeon se potrivesc aproape fără nici un obstacol, deoarece sunt foarte bine testate. Unele probleme au apărut în reinstruirea profesională a celor din diferite departamente, deoarece în multe cazuri a fost necesar să își reconsidere abordarea proceselor respective, iar în unele cazuri a fost necesar să se reorganizeze responsabilitățile și fluxul de muncă. Părți importante ale documentației care se refereau la aceste procese definitiv modificate au fost trecute cu vederea, deci a fost nevoie de un efort de recuperare pentru a nu afecta termenul proiectului.

XML este încă în dezvoltare, deci s-au creat planuri pentru orice eventualitate care să asigure un liant provizoriu în cazul în care soluțiile propuse nu funcționează. Iar migrarea finală a aplicației de la producție, care a pornit totul, a fost întârziată, între timp sistemele funcționând în paralel din cauza problemelor neprevăzute în migrarea de la CA-Manman Classic la CA-Manman/MK, datorate întârzierilor înregistrate în lansarea noii versiuni CA. Nimeni nu este perfect și nici un proiect important nu se realizează integral conform planurilor.

Sisteme ERP actuale

În secțiunile următoare vom studia în detaliu câțiva furnizori, dar mai întâi, iată o listă a furnizorilor actuali de linii de produse ERP și MRP. Unii dintre aceștia sunt discutați separat în secțiunile următoare:

Notă MRP (*Manufacturing Resource Planning* – planificarea resurselor de producție), acronim care pare să nu mai fie la modă acum, este precursorul ERP, care a extins sfera metodologiei și în domenii care nu sunt legate de producție.

- **SAP** – linia de produse SAP R/3 este, poate, cea mai cunoscută în lume la ora actuală. Situl lor Web are adresa `http://www.sap-ag.de/`.
- **BAAN** – BAAN asigură un set de produse pentru ERP și pretinde că acesta oferă o funcționalitate sporită și depășește ERP tradițional. Situl lor Web are adresa `http://www.baan.com/`. Unele browsere proiectate pentru persoanele cu handicap de vedere nu pot accesa situl Web BAAN. BAAN folosește din greu fișiere PDF Adobe, care au și ele propriile probleme de accesibilitate. Așezarea în pagină a fișierelor PDF pe care le-am văzut pare destinată monitoarelor de 21 de inci, deoarece textul este mult prea mic pentru a fi citit pe monitoare mai mici în fereastra afișată de situl Web.
- **PivotPoint** – PivotPoint este acum parte din Mapics, un furnizor de MRP menționat în următorul element al listei. Puteți accesa situl Web PivotPoint la adresa `http://www.pivotpoint.com/`. Unele browsere proiectate pentru persoanele cu handicap de vedere nu pot accesa situl Web PivotPoint.
- **Mapics** – Mapics este un furnizor de sisteme MRP tradiționale, cu extensii și seturi de aplicații care depășesc într-o oarecare măsură MRP tradițional. Prin achiziția PivotPoint, și-au extins linia de produse, incluzând și ERP complet. Puteți accesa situl Web Mapics la adresa `http://www.mapics.com/`. Deși situl Mapics este destul de accesibil, sunteți purtat printr-o serie enervantă de ecrane pentru a vă face să vă uitați la ecranul introductiv Macromedia Flash, o extraordinară pierdere de vreme.
- **Oracle** – Deși Oracle este cunoscut mai ales pentru tehnologia sa de baze de date relaționale, are disponibil un set complet de produse ERP. Vizitați situl Web Oracle la adresa `http://www.oracle.com/`. Unele browsere proiectate pentru persoanele cu handicap de vedere nu pot accesa situl Web Oracle. Oracle oferă accesibilitate, dar și un mesaj ironic care îi spune utilizatorului să își modernizeze browserul, o sugestie nepotrivită în orice context.
- **Manex Systems** – Manex Systems asigură un sistem ERP orientat în special spre producătorii de componente electronice. Puteți accesa situl Web Manex Systems la adresa `http://www.manex.com/`.
- **ERP Hub** – Acest sit este o sursă neutră față de furnizori pentru firmele care caută soluții ERP și o sursă bună de opinii mai mult sau mai puțin imparțiale. Puteți accesa situl Web ERP Hub la adresa `http://www.erphub.com/`.

Majoritatea acestor companii vor *într-adevăr* să achiziționați o soluție radicală și să aruncați la gunoi toate instrumentele existente legate de domeniul lor de expertiză, ceea ce poate să necesite analize ale proceselor afacerii, analize ale procedurilor de execuție și reinstruirea profesională a angajaților, la o scară mai mare decât ceea ce poate suporta la un moment dat firma dumneavoastră. Însă un element cheie al sistemelor mai bune este că pot să conlucreze cu sistemele existente și cu sistemele furnizorilor și clienților dumneavoastră, permițând astfel o apropiere treptată de scopurile ERP. Cu alte cuvinte, vă permit să vedeți beneficiile create prin investiție și nu vă cer să aruncați copilul o dată cu apa din scăldătoare.

SAP R/3, de exemplu, oferă o bibliotecă extinsă de instrumente create pentru a permite comunicarea între bazele de date moștenite și SAP R/3. Unii producători sunt specializați în asigurarea interfețelor cu SAP R/3, cum ar fi HAHT Software, care asigură o interfață Web pentru aplicațiile și bazele de date SAP R/3. Vizitați situl HAHT Software la adresa `http://www.haht.com/`.

Mulți dintre acești furnizori sunt capabili să asigure facilități și personal, pentru a vă permite să satisfaceți cerințele ERP în afara întreprinderii dumneavoastră. Aceste servicii sunt o bună modalitate de a gestiona riscurile implicate de ceea ce, pentru majoritatea domeniilor și firmelor, este o tehnologie care trebuie încă să își dovedească puterea. Unele implementări mai vechi au fost dezastruoase și a trebuit să fie abandonate; dacă angajați furnizorul în acest proces, veți avea mai multă putere dacă lucrurile merg prost și veți fi mai puțin expus dacă proiectul eșuează.

Pe de altă parte, dezvoltarea unei componente cheie a planurilor strategice în afara întreprinderii poate fi văzută ca o abdicare de la responsabilități. Abilitățile dezvoltate în afara întreprinderii pot să fie mai slabe, din cauza lipsei de experiență în interiorul organizației, deteriorând astfel deprinderile și baza de cunoștințe ale firmei. De asemenea, asigurați-vă că prin dezvoltarea proiectului în afara organizației nu vă fixați la o soluție care acum este ultimul răcnet, dar în câțiva ani devine o soluție patetică și lipsită de eleganță. ERP este încă la început și atât dezvoltarea în afara firmei (cu un contract pe termen lung) cât și cumpărarea au dezavantajul că vă blochează într-o soluție pe care poate că o veți regreta mai târziu.

Orice soluție alegeți, asigurați-vă că aceasta are un drum clar de modernizare o dată cu evoluția tehnologiei. Doar nu doriți să rămâneți cu Mototol 1.1 în timp ce restul lumii folosește Fulger 2000, numai din cauză că furnizorul a renunțat la un sistem mai vechi în favoarea unuia nou.

SAP R/3

Unul dintre cele mai vestite sisteme de întreprindere pentru proiectanți și integratori este SAP AG, al căror produs R/3 este folosit pe scară largă în companiile importante

pentru a crea seturi de aplicații modulare folosind trusa extinsă de componente modulare R/3. Integrarea diferitelor componente într-un sistem complet este obișnuită pentru SAP. De fapt, R/3 este conceput ca o serie de unități modulare care se pot conecta și care asigură unități distincte ale logicii întregii afaceri. Aceste unități care se pot conecta sunt combinate în „seturi" care cuprind toate activitățile întreprinderii.

Prin această abordare de meniu de pizza a ERP, SAP AG se obligă să minimizeze dezmembrarea și integrările particularizate costisitoare, deoarece toate componentele sunt proiectate de la început să conlucreze. Acestea dispun și de interfețe destul de bune cu aplicațiile moștenite, deci în majoritatea cazurilor nu trebuie să faceți totul dintr-o dată, deși pot fi necesare mici codări particulare. O companie poate alege dintr-o listă vastă de aplicații gata create pentru a implementa cerințele proprii, principala grijă fiind costul, și nu durata de dezvoltare sau chiar succesul proiectului. Eliminarea acestor riscuri are costul său, deoarece modul în care operează SAP R/3 poate să nu corespundă perfect cu cerințele dumneavoastră.

Porțiunile în care SAP va folosi XML cuprind containere de date pentru apelarea Business API (BAPI) al R/3 și alte interfețe de programare. SAP intenționează să asigure suport XML ca format de conținut pentru formularele electronice, cum ar fi comenzi de achiziție trimise prin Internet în combinație cu cataloage on-line și XML ca și container de date pentru documentele afacerii, informațiile de flux de operații și rapoarte.

SAP intenționează să asigure XML pentru toate interfețele sale majore, să stabilească XML ca un format folosit pe larg în întreaga sa linie de produse și să furnizeze numeroase produse, inclusiv aplicația SAP Business to Business Procurement, care exploatează XML ca mecanism de transfer. Aruncând o privire peste anunțurile din ziarele importante, veți vedea că programatorii de SAP R/3 sunt foarte căutați, iar combinația XML – un alt element fierbinte în CV-uri – și SAP R/3 este probabil una dintre cele mai bune mișcări în carieră.

PeopleSoft

PeopleSoft utilizează Open Integration Framework (OIF – cadrul de integrare deschisă) pentru a asigura interfața atât cu Internetul, cât și cu XML ca mediu de transfer. Deși programatorul mai are de creat mult din logica interfeței, stabilitatea mecanismului de transfer XML presupune că, după ce logica interfeței inițiale este realizată, evoluția ulterioară poate fi asigurată prin transformările XSL ale interfeței existente. Este mult mai ușor să preluați un flux XML și să îl transformați în altul decât să descifrați un flux de date binare, care pot fi comprimate și prescurtate în maniere particulare, și să îl metamorfozați în altul la fel de obscur.

BAAN

BAAN asigură mai multe niveluri de soluții pentru afaceri, de la relații cu clienții la ERP pentru administrarea de lanțuri de firme. Deși pentru integrarea strânsă și pentru suportul eforturilor mai vechi de integrare sunt utilizate API-uri mai mult sau mai puțin tradiționale, XML este o parte a strategiei generale BAAN, îndreptat spre ceea ce BAAN numește Business Process Level (nivelul de procese al afacerii) și Business Community Level (nivelul comunității de afaceri).

Soluția ERP integrală la momentul actual se numește BaanERP. BAAN oferă și un produs numit Baan-on-Board, care leagă atât software-ul cât și hardware-ul într-un mediu complet echipat și gata de funcționare, care duce la un alt nivel simplificarea ERP. Desigur, există costuri de calificare pentru a aduce angajații existenți la nivelul cerut de noul mediu și poate fi o idee bună să angajați personal calificat pentru noul mediu care să încorporeze abilitățile interne ale întreprinderii.

Conversii de sisteme

În secțiunea anterioară „Comunicarea între sistemele de întreprindere", ați văzut ce probleme apar în cazul modernizării unui sistem existent cu interfețe pentru diferite sisteme din interiorul întreprinderii sau ale furnizorilor și clienților existenți. Aceleași probleme apar și în cazul migrării la un sistem complet nou. În orice migrare, va exista o perioadă în care sistemele vor lucra în paralel pentru a se confirma faptul că noul sistem efectuează aceleași operații ca și sistemul vechi, și cu aceeași precizie. Înainte, această abordare presupunea angajarea de echipe duble care introduceau date în ambele sisteme, deoarece era de la sine înțeles că cele două sisteme nu puteau comunica. Însă dacă ambele sisteme înțeleg XML, sau sunt învățate să îl înțeleagă, atunci același flux de date poate lucra cu ambele sisteme, reducându-se astfel posibilitatea erorilor sau măcar reducându-se costurile de personal.

Adăugarea ERP la întreprindere: ajutorul XML

Pentru a ilustra problemele care stau în fața unei întreprinderi care adaugă ERP la operațiile sale, să studiem cazul Grupului de Asamblare de Dispozitive Catamount. Aceștia au o problemă: apar tot timpul fluctuații necontrolate de inventar, deoarece furnizorii nu pot să prevadă cantitatea de piese care trebuie asigurată și așteaptă mai întâi comanda de la Catamount. Timpul pentru producerea pieselor este destul de lung, iar Catamount așteaptă până când sunt destule ca să umple un vagon și să economisească astfel din costurile de transport. Astfel, furnizorii pot să aștepte săptămâni până la următoarea comandă, iar apoi să li se ceară dintr-o dată zece mii de piese. Adesea nu pot să satisfacă aceste comenzi. Au și ei problemele lor de inventar și nu pot să-și permită să păstreze zece mii de piese doar pentru eventualitatea că apare comanda Catamount pentru încărcarea vagonului. Inventarul se efectuează manual, o dată pe săptămână.

Soluția: furnizorii trebuie să vadă zilnic starea actuală a inventarului de piese. Văzând ritmul în care scad stocurile, ei pot să estimeze destul de bine momentul în care va apărea următoarea comandă. De fapt, pot chiar să expedieze un vagon la momentul potrivit, bazându-se pe tendințele actuale, dacă totul este aranjat în termeni contractuali. Catamount trebuie să își automatizeze sistemul de inventariere, deoarece rapoartele săptămânale nu sunt suficiente pentru a observa din timp tendințele. Însă, prin introducerea unui control bazat pe cititori de coduri de bare plasați la ieșirea din depozit, fiecare dispozitiv care iese pe ușa depozitului este înregistrat automat.

Deoarece Catamount folosește un sistem de inventariere învechit, dar care își face perfect treaba, s-a presupus că este cel mai logic să se scrie un modul care să traducă rapoartele inventarului într-un limbaj de inventariere bazat pe XML. Aceste rapoarte pot fi transmise periodic furnizorilor, iar aceștia pot cere rapoarte mai dese, dacă este necesar. Ca efect secundar, s-a descoperit că aceleași formulare de date pot fi folosite de cei de la contabilitate pentru a afla în ce stare se află stocurile din depozit, având astfel posibilitatea de a controla mai îndeaproape și în orice moment costul vânzărilor și capitalul fixat în inventar.

Catamount a decis să folosească o schiță de standard de la Electronic Business XML Initiative, cu adresa `http://www.ebxml.org/`, care încearcă să unifice multitudinea de standarde pentru e-business bazate pe XML într-un standard internațional, proiectat pentru a crea o piață mondială unitară. Proiectul este în curs de desfășurare.

Un exemplu de problemă similară din lumea reală poate fi găsit la MRO.com (`http://www.mro.com/`), care furnizează software bazat pe XML pentru a ușura comunicațiile dintre partenerii comerciali. Produsele MRO.com sunt folosite de firme cum ar fi Grainger, al cărei catalog cuprinde informații despre mai mult de 2000 de furnizori.

Printre alți producători care oferă sau intenționează să ofere software similar sunt și Ariba Technologies, Clarus Corporation, Commerce One, Intelisys și TRADEX Technologies.

Depanarea sistemelor ERP

ERP sau funcţionează, sau nu

ERP este simplu din punct de vedere conceptual: se schimbă fluxuri de date și ambele părți le înțeleg. Posibilitățile de eșec sunt reduse, având în vedere natura guralivă a lui XML. Puteți vedea efectiv datele intrând și ieșind în timp real, deci depanarea procesului care inițiază și păstrează aceste fluxuri de date se rezumă la a examina fluxurile și a verifica dacă datele trimise nu sunt gunoi, iar cele primite nu conțin erori. Deoarece XML se bazează pe text și este transparent prin design, posibilitățile de eroare sunt minime.

Examinați problema. Iată o abordare pas cu pas pentru a o rezolva:

1. Verificați fluxul de date de ieșire. Este conform cu specificațiile?
2. Dacă da, treceți la pasul 4.
3. Anunțați programatorul să vină să rezolve problema. Treceți înapoi la pasul 1.
4. Există trafic de intrare? Dacă da, treceți la pasul 6.
5. Conexiunea este întreruptă? Dacă nu, treceți la pasul 7.
6. Anunțați inginerul de rețea să rezolve problema. Treceți înapoi la pasul 1.
7. Fluxul de date de intrare este corect?
8. Dacă nu, anunțați programatorul furnizorului. Treceți înapoi la pasul 1.
9. Treceți înapoi în pat.

Acesta este, în câteva cuvinte, procesul și este foarte simplu. O mare parte din această simplitate provine din transparența mecanismului de transfer al XML, deoarece nu aveți nevoie de tabele complexe și de un software de afișare special pentru a examina câmpurile din fluxul de date.

Partea a IV-a

Alte aplicaţii ale XML

21 RDF (The Resource Description Framework – cadrul descrierii resurselor)
22 XHTML în detaliu
23 Utilizarea SMIL (The Synchronized Multimedia Integration Language – limbajul de integrare multimedia sincronizat)
24 Utilizarea MathML (The Math Markup Language – limbajul de marcare matematic)
25 Accesibilitate
26 Cooperarea în domeniu și resurse
27 Concluzii – viitorul în Web

CAPITOLUL

21

RDF

(The Resource Description Framework – cadrul descrierii resurselor)

În acest capitol

Metadate
Necesitatea unui limbaj de descriere separat
Definirea vocabularelor RDF universale
Dublin Core
PICS
Explorarea sintaxei RDF
Când trebuie adăugat RDF la XML
Depanarea RDF

Metadate

Metadatele se referă la datele *despre* date și se găsesc în forme diferite pe Web. Într-un fel, chiar HTML este metadată deoarece descrie textul și celelalte date conținute în pagină. Totuși, termenul se referă de obicei la datele care rămân în codul HTML sau XML după extragerea datelor efective. *PICS*, *Dublin Core* și *RDF* sunt limbaje de metadate sau vocabulare, în funcție de punctul dumneavoastră de vedere, care permit descrierea în alți termeni a documentelor Web și a elementelor care le alcătuiesc.

Oricine a făcut vreodată experiențe cu un motor de căutare a fost martorul enormei cantități de date care se află pe Web. Numărul de potriviri pentru multe căutări este de ordinul milioanelor, un număr de referințe pe care nimeni nu le poate verifica la un moment dat. Chiar dacă ignorăm efectul dăunător al spamming-ului – o practică prin care comercianții și autorii de reclame fără scrupule își împănează paginile Web cu multe cuvinte fără nici o legătură, pentru ca pagina să fie găsită ca răspuns în aproape orice căutare – problema este gravă.

Unii cred că XML va rezolva toate problemele asigurându-ne în mod magic căutări de informații în context, astfel încât `<firma>Cojocar S.A.</firma>` să fie ușor de deosebit de `<individ>Cojocar, Vasile</individ>` sau de `<meserie>Cojocar, categoria I</meserie>`. Multă lume nu se gândește dincolo de această observație.

Oricum, XML a fost creat în special pentru a încuraja dezvoltarea de vocabulare potrivite pentru diferite domenii, în orice limbă națională. DTD-urile pot fi create în orice limbă din lume, deci `<firma>Cojocar SRL</firma>` poate fi ușor de găsit pentru un vorbitor de limba română, dar `<gesellschaft>Cojocar GmbH</gesellschaft>` nu prea.

Într-o lume în care traducerea prin calculator ușurează din ce în ce mai mult depășirea barierelor de limbă, faptul că o companie are sediul în Germania sau în Japonia o face să fie nepotrivită pentru afaceri? Aproape oricine va fi de acord că nu așa stau lucrurile.

În orice caz, este necesar un vocabular universal care să permită etichetarea fără ambiguități a unor elemente din pagini, indiferent de limba vorbită și de etichetele pe care le-ați proiectat în DTD-urile dumneavoastră. De o bună bucată de vreme, acest vocabular este Dublin Core, care este în special orientat spre biblioteci, sau ceva similar. În ultimii ani, mecanismului de bază Dublin Core i s-a adăugat PICS, un standard care la început avea simplul scop de a eticheta siturile sigure pentru copii. În orice caz, ambele folosesc metode care nu se încadrează în strategia XML, dar oricum le vom studia în acest capitol pentru a ne face o idee despre utilitatea vocabularelor.

▶ Pentru mai multe informații despre vocabularul Dublin Core, **vezi** „Dublin Core", la **pagina 465.**

▶ Pentru mai multe informații despre standardul PICS, **vezi** „PICS", la **pagina 470.**

Acest capitol are mai puține ilustrații pe lângă cod, deoarece tratează metadatele, care prin definiție sunt ceva intangibil și metafizic și, dacă au efecte vizibile, acestea sunt puține. A încerca să vizualizați metadatele este ca și cum ați încerca să fotografiați un calmar uriaș pe fundul oceanului. Știți foarte bine că aceștia există și înoată tăcuți undeva în adâncurile întunecate ale oceanului, dar fotografierea unuia este o altă mâncare de pește.

RDF în acest context

RDF este o încercare de potrivire a abundenței de limbaje descriptive pentru metadate într-un cadru bazat pe XML (de unde îi vine și numele) care permite descrierea aproape oricărei scheme de etichetare în termenii unor triplete dintr-un graf orientat. RDF moștenește toate conceptele XML care vă permit să specificați proprietățile unui element. Schema limbajului său asigură duplicate pentru multe din facilitățile XML Schema, deci codările sunt ușor de manipulat. RDF este orientat spre obiecte, deși într-un mod mai greu de înțeles, deci are și avantajele specifice abordării orientate spre obiecte.

Notă Orice resursă poate fi privită ca un obiect, în sensul orientării spre obiecte. O resursă RDF reprezintă o instanță a unei clase RDF Schema. Perechile RDF proprietate/valoare acționează asupra acestor resurse fără să afecteze proprietățile pe care le moștenesc de la clasa lor de bază.

Deoarece RDF se bazează pe XML, orice browser cu suport XML poate să manipuleze RDF, fără procesări speciale, camuflări de comentarii sau elemente `CDATA` pentru a-l

ascunde vederii. RDF conţine multe căi care facilitează procesarea automată a resurselor Web, cum este cea realizată de diversele motoare de căutare. RDF mai poate fi folosit în multe alte moduri.

De exemplu, specificitatea încorporată în anumite vocabulare de metadate poate să asigure căutări mai bune pentru motorul dumneavoastră preferat. În biblioteci, faptul că este capabil să traducă sau să includă mai multe sisteme de catalogare asigură un acces mai bun la materialele stocate în diferite biblioteci din întreaga lume, o adevărată binefacere pentru ştiinţă şi cercetare. Permiţând agenţilor software inteligenţi să cutreiere planeta în căutarea unor resurse care corespund anumitor criterii, RDF poate facilita cercetările de orice tip.

RDF şi PICS pot să uşureze viaţa pentru părinţi sau alte persoane care caută sau vor să evite situri cu sau fără un anumit conţinut şi să prevină în acelaşi timp incidentele neplăcute. Permiţând afirmarea simplă a drepturilor de autor pentru un întreg sit, RDF poate simplifica sarcinile unui proiectant Web. Iar dacă este folosit împreună cu P3P pentru a exprima preferinţele de confidenţialitate ale unui utilizator, conjugate cu politica de confidenţialitate a unui sit Web, oamenii vor fi mai bine protejaţi împotriva unui marketing agresiv şi împotriva tehnicilor indiscrete. În sfârşit, RDF şi semnăturile digitale pot alcătui soluţia pentru construirea unui „Web al încrederii" care să încurajeze comerţul electronic şi alte aplicaţii sensibile la securitate.

Necesitatea unui limbaj de descriere separat

XML este proiectat pentru a permite oricui să facă aproape orice îi trece prin cap cu numele de elemente. Proiectanţii de DTD-uri pot crea etichete care au un sens şi sunt uşor de înţeles în limba lor sau în domeniul lor de activitate, deoarece lucruri identice pot fi văzute complet diferit în domenii diferite. Acelaşi individ poate fi `<pacient>Sarah Jones</pacient>` pentru doctorul său, `<student>Sarah Jones</student>` pentru şcoala la care învaţă, `<goruchwyliwr>Sarah Jones</goruchwyliwr>` ca supraveghetor la serviciu, în Ţara Galilor, şi `<contribuabil>Sarah Jones</contribuabil>` pentru Ministerul de Finanţe. Un singur nume nu poate să se potrivească pentru toate aceste sensuri, şi de altfel nici nu trebuie. Însă dacă datele sunt prezentate pe Web, ele pot fi utilizate pentru orice scop. Creatorul original al datelor poate să dorească să păstreze numele elementelor sale pentru propriile scopuri, dar altcineva poate să folosească alte etichete pentru aceleaşi date. RDF permite acest tip de translaţii.

RDF nu este o schemă generală de translatare a schemelor. Sintaxa, deşi destul de succintă având în vedere scopul propus, este mult prea bogată pentru a fi folosită în fiecare etichetă, chiar având cele mai bune intenţii. O pagină în care fiecare etichetă este însoţită de informaţiile RDF alternative ar fi ridicol de încărcată. Întreţinerea unor astfel de pagini ar fi un coşmar, în afară de cazul în care ar fi generate automat.

Ceea ce realizează RDF foarte bine este să asigure un instrument limitat ce permite asocierea unei cantităţi limitate de metadate la o pagină, într-un mod transparent pentru utilizator.

RDF indică înţelesul datelor

Căutarea unui anumit lucru pe Web poate fi o experienţă frustrantă şi cu pierdere de vreme. Dacă ştiţi destule pentru a construi ceea ce doriţi, adică un şir de căutare care va

găsi comoara în grămada de balast numită Web, atunci poate că nici nu aveţi nevoie de informaţia respectivă. Este ceva foarte obişnuit să consumaţi o grămadă de timp, uneori ore, încercând să reperaţi o anumită informaţie. Asigurând un context, RDF vă permite să căutaţi informaţiile relevante, şi nu aiureli aleatoare.

De exemplu, fragmentul XHTML `<h1>Ford</h1>` este destul de obscur. Nu ştiţi dacă se referă la automobile, la centrul Betty Ford, la vadul unui râu sau la altceva. În schimb, în momentul în care ataşaţi adresa unei resurse care există efectiv pe Web printr-o descriere de resursă RDF, devine mult mai clar şi, în plus, oferiţi informaţii de fond pentru cei care doresc să studieze subiectul mai îndeaproape:

```
<rdf:Description
  about="http://www.whitehouse.gov/WH/glimpse/presidents/html/gf38.html">
  <h1>Ford</h1>
</rdf:Description>
```

Ford se referă evident la Gerald Ford, fostul preşedinte al SUA, iar referinţa Web este legată de o locaţie efectivă unde puteţi găsi mai multe informaţii despre el. Veţi afla mai multe despre sintaxa RDF în continuarea acestui capitol.

Să ne descurcăm în căutările fără context

Până când RDF va deveni disponibil peste tot pe Web, majoritatea căutărilor trebuie să fie restrânse cu ajutorul unor cuvinte cheie care s-ar putea să devină în final cuvinte cheie RDF. Deci dacă învăţaţi să căutaţi folosind instrumentele actuale, vă va fi mai uşor atunci când descrierile RDF vor fi banale.

Sugestie de la LeeAnne.com Words to weave by

În timp ce aşteptaţi să apară descrierile universale RDF, învăţaţi să utilizaţi caracteristicile avansate ale motorului de căutare fără ajutorul RDF. Deşi nu sunt nici pe departe perfecte, interogările din cuvinte bine alese vă permit să sugeraţi o parte a contextului care va fi clarificat prin proiectarea XML RDF.

Multe din căutările pe care le faceţi sunt lipsite de context. Puteţi încerca să furnizaţi un fel de context alcătuind instrucţiuni de căutare complicate, care încearcă să reducă numărul de potriviri de la milioane, la un număr rezonabil. De exemplu, să presupunem că doriţi să vedeţi ce fel de combină muzicală să vă cumpăraţi, deci încercaţi o căutare cu `+"Stereo System"` în AltaVista. Veţi obţine ceva mai mult de 4000 de rezultate. Ar lua mult timp să vă uitaţi prin toate. Observaţi că multe dintre situri sunt în Geocities, un loc în care nu prea veţi găsi sfaturi bune, deci veţi elimina aceste intrări prin: `+"Stereo System" -geocities`. Rezultatele căutării scad doar cu vreo 200 de situri. Căutarea dumneavoastră nu este suficient de specifică.

Vă mai gândiţi puţin. Doriţi sfaturi pentru consumatori, nu? Veţi mai adăuga ceva la şirul de căutare: `+"Stereo System" -geocities +consumer`. Numărul de rezultate scade sub o sută, deci puteţi spera să vă găsiţi combina potrivită înainte de sfârşitul anului. Toate aceste jonglerii au fost necesare deoarece căutarea este lipsită de context. Nu puteţi identifica datele din sit cu nici o schemă asociată de metadate.

Dar cum ar fi dacă agentul dumneavoastră de căutare ar cunoaşte un standard de metadate (în prezent imaginar) care marchează paginile cu sfaturi pentru consumatori cu etichete special proiectate pentru a se adresa problemelor consumatorilor, cum ar fi costul, reparaţiile, calitatea sunetului, evaluarea generală şi aşa mai departe. În momentul de faţă, dacă tastaţi „What's the best stereo system to buy?" (Care este cel mai bun sistem

stereo de pe piaţă?) veţi obţine cam 44 de milioane de rezultate. Însă, dacă agentul ar fi doar cu puţin mai deştept şi şi-ar da seama că sunteţi în căutare de sfaturi despre sisteme audio, ar putea să iniţieze o căutare contextuală şi să producă o listă scurtă în câteva momente.

Pentru asta a fost creat XML, în particular RDF. Prin definirea unor vocabulare descriptive speciale, datele pot fi clasificate după sensul lor, şi nu după formă. Deci căutarea pe care am rafinat-o îndelung mai sus ar putea fi simplificată cu o sintaxă deocamdată imaginară: `+"Stereo System" +RDF:consumerInformation` şi ar putea avea ca rezultat chiar mai puţine potriviri. Reţineţi că în acest exemplu se presupune că cineva, posibil o asociaţie industrială sau o organizaţie de informare a consumatorilor, a definit un vocabular RDF care include `consumerInformation`, iar motorul de căutare folosit ştie că descrierile RDF au un înţeles special. Nu este ceva greu de imaginat, din moment ce majoritatea motoarelor de căutare actuale permit căutări restrictive după anumite etichete HTML.

Definirea vocabularelor RDF universale

Într-un fel, metadatele universale sunt un vis imposibil. Pentru orice schemă descriptivă pe care v-o puteţi imagina, există cineva care are o idee complet diferită despre ceea ce reprezintă resursa respectivă. De la cherestegiul care vede într-un copac metri cubi de scândură la ecologistul pentru care acelaşi copac este habitatul bufniţei pătate, comunităţi diferite pot conferi aceleiaşi resurse înţelesuri complet diferite. Metadatele funcţionează cel mai bine în grupuri clar definite, ai căror membri comunică frecvent şi pot să se înţeleagă asupra unui vocabular comun şi asupra unor subiecte de discuţii comune. Însă metadatele nu mai funcţionează aproape deloc dacă trebuie să treacă graniţe de orice fel, deci nu vor alcătui niciodată o rezolvare completă. RDF nu va permite interogări ca în Star Trek, în care întrebaţi numele mamei lui Ghilgameş, iar calculatorul trece la traducerea unui material pe care nu l-a mai văzut niciodată, direct de pe anticele tăbliţe de lut sumeriene, pentru a găsi răspunsul. RDF nu este un astfel de instrument.

Sugestie de la LeeAnne.com Words to weave by

În cazul în care căutaţi un anumit subiect, merită să petreceţi puţin timp căutând o organizaţie interesată de acelaşi subiect. Poate că organizaţia respectivă a rezolvat deja căutarea dumneavoastră şi a creat descrieri de metadate care vă pot ajuta.

În orice caz, dacă sunteţi un bibliotecar care încearcă să dea de *Genfremstilling af det gamble babyloniske Gilgamesh-epos om venneparret Gilgamesh og Engidu*; København, Nyt nordisk forlag, 1940, într-o bibliotecă de la capătul continentului, probabil în Danemarca sau Norvegia sau cine ştie unde, Dublin Core a fost creat tocmai în acest scop.

▶ **Vezi** „Dublin Core", la **pagina 465** pentru explicaţii despre funcţionarea Dublin Core şi despre redefinirea simplă a acestuia ca vocabular RDF.

Dublin Core este invenţia unui bibliotecar şi foloseşte un vocabular englez pentru a descrie datele care se înregistrează de obicei într-un catalog de fişe al unei biblioteci. Pentru oamenii de ştiinţă sau pentru cercetători, o intrare Dublin Core este un instrument inestimabil pentru identificarea documentelor. Şi, de fapt, multe motoare de căutare sunt deosebit de atente să indexeze datele găsite în câmpurile Dublin Core conţinute în documentele HTML.

În schimb, multe motoare de căutare nu folosesc aceste câmpuri din cauza potenţialului deosebit pentru înşelătorie din partea indivizilor fără scrupule, care mint în mod deliberat cu privire la conţinutul paginii, pentru a vă atrage pe siturile lor. Astfel, o căutare după expresia „corn silage research" (cercetare porumb furajer) vă va conduce nu numai la articole despre creşterea animalelor, care îi pot interesa pe fermieri, ci şi la un sit pornografic care foloseşte o redirecţionare client-side pentru a vă duce la o pagină de gust îndoielnic. Din moment ce intrarea de index care vă duce la această pagină este o minciună sfruntată, este destul de clar că cineva cu o inteligenţă normală se va simţi jignit de acest tip de deviere venală. Tocmai din cauza acestei probleme, multe persoane nu au încredere în motoarele de căutare şi refuză să le folosească, preferând să navigheze prin portaluri de încredere care, deşi ascund cea mai mare parte a Web-ului, măcar nu oferă surprize neplăcute.

Dar, în primul rând, celor mai mulţi nu le pasă de indexări. Nu mulţi îşi cataloghează colecţia de cărţi şi nici nu îşi pun înregistrările pe Web, excepţie făcând bibliotecarii. Poate cartea căutată este în clădirea vecină, în mii de exemplare, şi nu veţi afla niciodată. Cei mai mulţi nu vor face *nimic* cu *nici una* dintre iniţiativele de metadate, decât dacă sunt forţaţi sau plătiţi. De exemplu, un magazin de desfacere poate să-şi eticheteze marfa astfel: `<BeanieBaby>Blakie the Bear</BeanieBaby>`, astfel încât colecţionarii pasionaţi de jucării să găsească marfa şi să îi crească preţul, însă pentru Gică de la drogheria din colţ nu prea contează. Iar din cauza potenţialului de abuz, schemele de etichetare RDF de încredere vor fi realizate şi supravegheate de terţi şi vor fi supuse unui control strict în cazul utilizării de către persoane particulare.

Deci, cel puţin în viitorul apropiat, RDF şi celelalte forme de metadate vor fi domeniul specialiştilor şi profesioniştilor care, prin definiţie, sunt plătiţi să asocieze metadatele cu lucrările lor, şi domeniul câtorva indivizi care vor să folosească RDF numai de-ai dracului.

Dublin Core

Dublin Core este încununarea unor ani de eforturi depuse de un grup de bibliotecari, informaticieni şi alţi membri interesaţi ai lumii publicaţiilor. Acesta descrie o înregistrare bibliografică ideală care poate fi folosită în diferite scopuri, inclusiv Union Library Catalogs (cataloagele uniunii bibliotecilor), motoare de căutare, extrageri de date şi altele. Deoarece datează de pe vremea lui HTML 2.0, a folosit la început un singur mecanism suportat de agenţii utilizatori primitivi ai HTML, un vocabular standard care trebuia folosit în etichetele meta.

Notă

Discuţii extinse şi referinţe despre istoria Dublin Core pot fi găsite pe situl Web al Online Computer Library Center, `http://www.oclc.org/` deoarece OCLC s-a implicat masiv şi chiar de la început în planificarea şi dezvoltarea standardului. OCLC găzduieşte organizaţia Dublin Core la adresa `http://purl.org/DC/` şi, de asemenea, Persistent Uniform Resource Locator (PURL) la adresa `http://purl.org/`.

Dacă nu ştiţi ce este o etichetă meta, aflaţi că este un element vid cu trei atribute care ne interesează în acest moment: `name`, `content` şi `scheme`. Completând câmpurile de nume şi conţinut puteţi instanţia orice vocabular într-o manieră care poate fi citită de calculator, iar atributul `scheme` vă permite să identificaţi vocabularul folosit. Deci

```
<meta name="Subject" content="literatura secolului 18">
```

identifică o resursă care are ca subiect literatura secolului 18, deși aici sintaxa Dublin Core nu este completă.

Chiar și în cadrul Dublin Core au apărut diferențe. Sintaxa pentru utilizarea Dublin Core, așa cum este specificată în RFC 2413, Dublin Core Metadata for Resource Discovery, este ușor modificată de motorul de căutare AltaVista. Elementul din Core "Subject" este modificat în "Keywords". Dar asta nu e totul – și sintaxa diferă puțin. Dublin Core folosește în HTML o versiune inițială a ceea ce va deveni zona de nume XML, aceasta fiind identificată de un element `link`, iar fiecare termen care aparține zonei de nume este identificat cu un prefix:

```
<link rel="schema.DC"
        href="http://purl.org/DC/elements/1.0/">
<meta name="DC.Subject" content="literatura secolului 18">
```

Sintaxa echivalentă redată în XHTML este

```
<html DC:xmlns="http://purl.org/DC/elements/1.0/">
<DC:meta name="Subject" content="literatura secolului 18" />
```

În RDF ar fi

```
<RDF DC:xmlns="http://purl.org/DC/elements/1.0/">
<DC:Subject>literatura secolului 18</DC:Subject>
```

În practică, mai ales din cauză că AltaVista și alte motoare de căutare ignoră această formă, identificatorul zonei de nume este omis sau duplicat cu o versiune în care zona de nume este omisă. Chiar și standardul a fost revizuit, deși RFC-ul nu a fost modificat. Versiunea curentă a standardului este 1.1, în care se adaugă câteva câmpuri ici și colo, și o puteți găsi la `http://purl.org/DC/`.

Dacă la toate acestea adăugăm și răspândirea utilizării etichetelor în alte țări, folosind limba națională, este de înțeles dezordinea care împiedică lizibilitatea. De exemplu, în Danemarca ar trebui să folosiți următorul tabel de corespondențe pentru a vă eticheta paginile:

Danish Concept	Dublin Core Element Name
Titel:	dc.Title
Udgave:	dc.Title.Release
Forfatter eller ophav:	dc.Creator
Emne og nøgleord:	dc.Subject
Beskrivelse:	dc.Description
Udgiver:	dc.Publisher
Anden bidragsyder:	dc.Contributor
Dato:	dc.Date
Ressourcetype:	dc.Type
Format:	dc.Format
Identifikator:	dc.Identifier
Kilde (originalt værk):	dc.Source
Sprog:	dc.Language
Parallel trykt udgave:	dc.Relation.IsVersionOf
Filiste:	dc.Relation.HasPart
Forudsatte programmer:	dc.Relation.Requires
Dækning:	dc.Coverage
Rettigheder:	dc.Rights

Observați că danezii au extins Dublin Core cu câteva valori proprii. Acest fapt ilustrează tensiunile dintre limbi și etichete și dintre eforturile de internaționalizare și caracterul practic. Oricine dorește etichete care au înțeles în limba sa. Dacă nu înseamnă nimic, nimeni nu le va putea folosi fără ajutor. Deși un danez recunoaște ușor conceptul de *limbă* în *sprog*, cum va recunoaște instantaneu, și mai ales cum va ortografia termenul într-o pagină Web fără un efort mental? Sistemul MARC, folosit de numeroase biblioteci naționale, inclusiv de Biblioteca Congresului SUA și Biblioteca Națională Canadiană, evită această problemă etichetând fiecare câmp cu un număr aleator. Prin urmare, nimeni nu mai găsește un înțeles pentru câmpuri fără să fie special pregătit, astfel că toată lumea este la egalitate în ceea ce privește etichetarea.

Având în vedere relativa omniprezență a englezei față de daneză, în mod superficial ar avea mai mult sens ca danezii să învețe puțină engleză decât să învețe daneză vorbitorii de engleză, dar engleza este foarte greu de învățat în multe regiuni ale lumii. Figura 21.1, de exemplu, vă prezintă Dublin Core făcut mai accesibil pentru vorbitorii de arabă. Folosește identificatorii în engleză, dar explicațiile sunt în arabă.

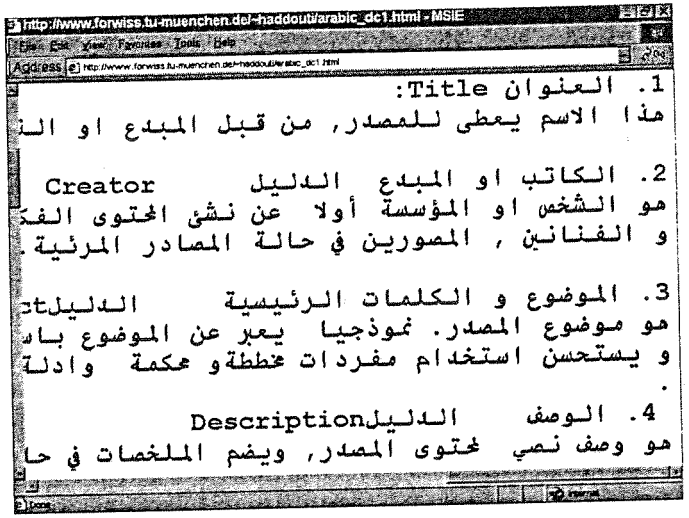


FIGURA 21.1
Dublin Core este descris aici în arabă.

RDF ușurează manipularea diferitelor limbi deoarece puteți folosi atributul `xml:lang` pentru a identifica diferite versiuni ale numelor de etichete care vor fi redate de diferite medii de creare:

```
<rdf:Description about="http://purl.org/dc#Subject">
  <rdfs:label xml:lang="en">Subject</rdfs:label>
  <rdfs:Comment xml:lang="en">The subject of the resource.</rdfs:Comment>
  <rdfs:label xml:lang="dk">Emne</rdfs:label>
  <rdfs:Comment xml:lang="dk">Ressourcens emne.</rdfs:Comment>
  <rdfs:label xml:lang="fr">Suject</rdfs:label>
  <rdfs:Comment xml:lang="fr">Le sujet de la ressource.</rdfs:Comment>
</rdf:Description>
```

Deci, pentru o audiență daneză, un autor danez ar putea folosi

```
<rdf …
    DC:xmlns="http://purl.org/DC/elements/1.0/">
    <DC:Emne>Bogføringskreds</DC:Emne>
</rdf>
```

în loc de

```
<rdf …
    DC:xmlns="http://purl.org/DC/elements/1.0/">
    <DC:Subject>Bogføringskreds</DC:Subject>
</rdf>
```

folosind aceeași zonă de nume multilingvă a Dublin Core. Translațiile între diferitele versiuni Dublin Core în variatele limbi acceptate pot fi realizate automat folosind XSLT. De fapt, dacă folosiți un mediu de creare bine ales, nici nu va mai fi nevoie de traduceri deoarece numele corect al etichetei în limba preferată a autorului poate fi afișat automat, deși editările efective sunt salvate în formatul simplu (în engleză) al sintaxei Dublin Core. Tabelul 21.1 vă prezintă valorile vocabularului de bază Dublin Core, împreună cu înțelesul asociat acestora.

Tabelul 21.1 Setul de elemente pentru metadate Dublin Core, versiunea 1.1

Numele elementului	Descriere
`Title`	Numele pe care îl poartă resursa în mod obișnuit, cum ar fi titlul unei cărți sau numele unui sit sau al unei pagini Web.
`Creator`	Persoana sau entitatea răspunzătoare în principal pentru conținutul intelectual al resursei, cum ar fi autorul sau proprietarul unei lucrări. Atenție, în practica mondială se poate cere identificarea creatorului uman al unor obiecte de artă, iar creatorul uman poate să păstreze anumite drepturi pe termen nelimitat în condițiile legilor internaționale.
`Subject`	Tema resursei. Adesea, subiectul este exprimat prin cuvinte cheie, fraze cheie sau coduri de clasificare ce descriu tema resursei. Cea mai bună soluție este selectarea unei valori dintr-un vocabular controlat sau dintr-o schemă de clasificare formală. Pe Web, acest câmp ar trebui să îl duplicați cu un element `"Keywords"`, pentru a se potrivi cu numeroasele servicii care depind de el sau îl așteaptă.
`Description`	O descriere a conținutului resursei. Aceasta poate conține un rezumat, un cuprins, o referință la o ilustrare grafică a conținutului sau o descriere liberă a conținutului, dar nu este limitată doar la aceste variante.
`Publisher`	Entitatea răspunzătoare pentru disponibilitatea resursei, cum ar fi editorul unei cărți sau entitatea responsabilă pentru un sit Web.
`Contributor`	O persoană sau o entitate răspunzătoare pentru contribuțiile aduse la conținutul resursei. De exemplu, poate fi o persoană, cum ar fi un ilustrator sau un redactor, o organizație sau un serviciu.
`Date`	O dată asociată cu un eveniment din ciclul de viață al resursei. De obicei, `Date` se referă la data creării sau la data primei publicări a resursei. Pe cât posibil, codarea datei trebuie realizată în formatul ISO 8601 și cu forma AAAA-LL-ZZ.
`Type`	Natura sau genul conținutului resursei. `Type` conține termeni care descriu în general categorii, funcții, genuri sau niveluri de structurare ale conținutului. `Type` ar trebui selectat dintr-un vocabular controlat, cum ar fi lista standard de tipuri Dublin Core: `collection`, `dataset`, `event`, `image`, `interactive resource`, `model`, `party`, `physical object`, `place`, `service`, `software`, `sound` și `text`. Pentru descrierea manifestării fizice sau digitale a resursei se folosește elementul `Format`.
`Format`	Manifestarea fizică sau digitală a resursei în lumea reală. `Format` poate să includă tipul de mediu sau dimensiunile resursei. `Format` poate fi folosit pentru a determina ce software, ce hardware și ce alte echipamente sunt necesare pentru a afișa sau a folosi resursa (de exemplu, un cititor de microfilme). Ca dimensiuni, de exemplu, mărimea – poate fi importantă pentru o carte dintr-o bibliotecă, deoarece cărțile de format mare sunt depozitate pe rafturi diferite – sau durata. Valorile pentru format ar trebui, pe cât posibil, să fie selectate dintr-un vocabular controlat, cum ar fi lista tipurilor de medii Internet (MIME) care definesc formatele pentru mediile informatice.
`Identifier`	O referință fără ambiguități la resursă într-un context dat. Dacă există un șir sau un număr care identifică resursa și se conformează cu un sistem formal de identificare, cum ar fi un număr de catalog, se poate folosi acesta. Dintre sistemele formale de identificare putem aminti Uniform Resource Identifier (URI), care include Uniform Resource Locator (URL); Digital Object Identifier (DOI); International Standard Book Number (ISBN) – ultimele două sunt Uniform Resource Names, sau URN-uri.
`Source`	O referință la o resursă din care a fost derivată resursa de față. Resursa de față poate fi derivată ca un tot sau numai parțial din resursa `Source`. Dacă există un șir sau un număr care identifică resursa `Source` și se conformează cu un sistem formal de identificare, se poate folosi acesta.
`Language`	Limba în care este scris conținutul intelectual al resursei. Valorile elementului `Language` trebuie codate așa cum este definit în RFC 1766, deci conțin codul limbii format din două litere preluat din standardul ISO 639, urmat opțional de codul țării format din două litere preluat din standardul ISO 3166. De exemplu, `en` pentru engleză, `fr` pentru franceză sau `en-UK` pentru engleza vorbită în Regatul Unit. În cazul unei cărți bilingve se vor folosi două elemente `Language` și așa mai departe.
`Relation`	O referință la o resursă înrudită. Dacă există un șir sau un număr care identifică resursa înrudită și se conformează cu un sistem formal de identificare, se poate folosi acesta.

Tabelul 21.1 (continuare)

`Coverage`	Extinderea sau orizontul conținutului resursei. `Coverage` include de obicei locația spațială (un nume de loc), perioada temporală (o etichetă de perioadă, o dată sau un interval de date) sau jurisdicția (cum ar fi o entitate administrativă denumită). Ar trebui să selectați, dacă este posibil, o valoare dintr-un vocabular controlat (de exemplu, folosiți Thesaurus of Geographic Names – dicționarul de denumiri geografice) și să utilizați, dacă se poate, nume de locuri și nume de perioade în locul identificatorilor numerici. Pentru conținutul Internet, această valoare este de obicei `"global"`, dar pentru o carte de adrese a unui oraș poate fi numele orașului, iar pentru rețeaua internă a unei corporații poate fi `"Company Confidential"`.
`Rights`	Informații despre drepturile deținute asupra resursei. Elementul `Rights` poate să conțină o declarație de administrare a drepturilor pentru resursă sau o referință la o locație sau la un serviciu care asigură această informație, cum ar fi ASCAP, BMI sau SESAC pentru înregistrări muzicale. Informațiile despre drepturi cuprind adesea drepturile de autor și alte drepturi de proprietate intelectuală sau de altă natură. Dacă elementul `Rights` este absent nu se pot face nici un fel de presupuneri asupra statutului acestor drepturi sau al altora în ceea ce privește resursa în cauză.

Pentru a demonstra cum se folosește Dublin Core într-un document XHTML, iată de exemplu un set de declarații Dublin Core pentru pagina introductivă a sitului meu:

```
<link rel="schema.dc" href="http://purl.org/DC/elements/1.0/">
    <meta name="DC.Title" lang="en" content="White Rose - LeeAnne.com" />
    <meta name="DC.Creator" content="Lee Anne Phillips" />
    <meta name="DC.Type" content="text" />
    <meta name="DC.Language" content="en" />
    <meta name="DC.Coverage" content="global" />
    <meta name="DC.Rights" content="Copyright (c) 1996 Lee Anne Phillips - All
Rights Reserved Worldwide" />
    <meta name="DC.Subject" lang="en" content="Lee Anne Phillips; consultant;
author; networking; telephony; project manager" />
    <meta name="DC.Description" lang="en" content="The Web site of White Rose and
LeeAnne.com Lee Anne Phillips Lee Ann Phillips" />
    <meta name="DC.Date" scheme="WTN8601" content="1999-07-15" />
    <meta name="DC.Source" content="D:\LeeAnne\index.html" />
    <meta name="DC.Format" content="text/html" />
    <meta name="DC.Identifier" content="http://www.leeanne.com/index.html" />
```

Observați că prefixele zonelor de nume respectă Warwick Framework, care a precedat cu câțiva ani efortul actual al W3C legat de zonele de nume.

PICS

PICS, Platform for Internet Content Selection – platforma pentru selecția conținutului Internet, a apărut ca răspuns la preocuparea părinților față de caracterul nepotrivit pentru copii al multor situri Web. Multe organizații de evaluare a siturilor fie solicită evaluări voluntare ale acestora, fie verifică efectiv conținutul siturilor și furnizează abonaților „liste sigure" de situri considerate potrivite pentru copii. Câteva dintre aceste situri sunt menționate în secțiunile următoare.

Sugestie de la LeeAnne.com Words to weave by

Atenție, multe browsere nu acordă nici o atenție etichetelor PICS. Dacă aveți un copil isteț, înlocuirea unui browser cu altul care lasă orice să îi treacă pragul este o joacă.

Evident, ambele sisteme sunt ușor de folosit în mod abuziv, deși din perspective diferite. Dacă un sit „adult" care satisface persoanele ce doresc să vadă poze cu necunoscuți dezbrăcați este evaluat ca fiind fără conținut sexual, acesta va trece prin unele filtre, deși mulți părinți ar avea obiecții față de accesul copiilor la acest sit.

Pe de altă parte, serviciile de evaluare restrictive elimină automat orice sit care conține cuvinte legate de femei, cum ar fi „breast" (sân) sau „menstrual". Astfel încât fetele pot fi împiedicate să acceseze situri de informații care sunt special concepute pentru ele și care rezolvă interesul lor natural pentru pubertate printr-un conținut potrivit vârstei, asigurat de profesioniști în sănătate.

Alții pot să fie preocupați de libertatea de exprimare și de libertatea cuvântului. Este foarte ușor pentru un serviciu de evaluare să elimine siturile cu a căror politică nu este de acord, deși au un conținut inofensiv. Deci PICS, care a pornit de la sine, poate fi cel mai bun compromis pentru cei mai mulți dintre noi. Iar flexibilitatea sistemului PICS permite folosirea acestuia pentru aproape orice criteriu de selecție, cu condiția ca lista de criterii să nu fie prea mare.

RSACi: implementarea unui vocabular PICS

Mai multe organizații furnizează liste de vocabulare PICS predefinite, una dintre cele mai importante fiind RSACi la `http://www.rsac.org/`.

Utilizarea sistemului RSACi este simplă; înscrieți-vă la situl Web, răspundeți la câteva întrebări despre nuditate, sex, violență și limbă și vi se oferă un șir de evaluare pe care puteți să îl inserați în pagina introductivă a sitului dumneavoastră, în secțiunea `<head>`.

Iată eticheta de evaluare a sitului meu Web:

```
<meta http-equiv="PICS-Label"
content='(PICS-1.1 "http://www.rsac.org/ratingsv01.html" l gen true comment "RSACi
North America Server" for "http://www.leeanne.com"
on "1999.11.03T14:41-0800" r (n 0 s 0 v 0 l 0))'>
```

O etichetă de evaluare de acest fel este generată în câteva minute. Observați că șirul de conținut este încadrat de apostrofuri și cuprinde subșiruri între ghilimele. Astfel, eticheta poate să aibă un conținut destul de arbitrar și să se potrivească totuși cu prototipul etichetei `<meta>`. Deoarece XHTML folosește aceeași sintaxă, putem presupune că o pagină XHTML nu va avea nevoie de un alt mecanism, deși există propuneri de a folosi RDF ca substitut.

Alte situri de evaluare PICS

RSACi nu este singura sursă de vocabulare PICS. Se pot menționa și alte situri de evaluare: Vancouver Webpages Rating Service, la `http://vancouver-webpages.com/VWP1.0/`, oferă categorii tipic canadiene care dezvăluie sărăcia intelectuală și morală a altor furnizori de

vocabulare PICS, care se concentrează pe vicii şi neglijează virtuţile. Aceştia au criterii de selecţie separate pentru Multiculturalism, Educational Content (conţinut educativ), Environmental Awareness (conştientizare ecologică), Tolerance (toleranţă), Violence (violenţă), Sex, Profanity (caracter profanator), Safety (siguranţă), Canadian Content (conţinut canadian), Commercial Content (conţinut comercial) şi Gambling (jocuri de noroc). Ca părinte, aş prefera să controlez toate aceste lucruri, din moment ce este foarte posibil ca un sit care susţine prejudecăţi nefaste şi intoleranţă să facă asta folosind cel mai frumos limbaj şi să treacă astfel de limitările RSACi. Iată evaluarea VWP pentru situl meu:

```
<META http-equiv="PICS-Label"
content='(PICS-1.1 "http://vancouver-webpages.com/VWP1.0/" l gen true comment
"VWP1.0" by "leeanne@leeane.com" on "1999.11.03T12:05-0800" for
"http://www.leeanne.com/"
 r (MC -2 Gam -1 Com 0 SF -1 Edu -1 S 0 Can -1 V 0 Env -1 P 0 Tol -2 ))'>
```

Mai există bineînţeles şi alţii: Adequate.com se concentrează asupra violenţei la `http://www.adequate.com/ratings/`, dar situl lor nu este prea bine realizat în acest moment. Pagina introductivă, de exemplu, este complet inaccesibilă pentru utilizatorii orbi şi este confuză pentru toţi ceilalţi. Poate ar trebui să îl vizitaţi dintr-un motiv sau altul.

Sugestie de la LeeAnne.com Words to weave by

Nu scăpaţi niciodată ocazia să vedeţi greşelile altora. Ca temă de casă, vizitaţi situl `http://www.adequate.com/` şi vedeţi dacă are vreo logică pentru dumneavoastră sau vi se pare uşor de folosit. Să sperăm că vor analiza cât mai curând şi cât mai serios designul şi îl vor reface. Nu există nici o scuză pentru un design incomod.

Safe for Kids

Situl Safe for Kids, la `http://www.weburbia.com/safe/ratings.htm`, are doar trei niveluri pentru o singură categorie – Safe for Kids (sigur pentru copii), Parental Guidance (sub supravegherea părinţilor) şi Adults Only (numai pentru adulţi) – având ca avantaj simplitatea, chiar dacă nu ţine cont de diferenţele de opinii legate de înţelesul categoriilor respective.

Safe Surf

Safe Surf, la adresa `http://www.safesurf.com/`, are categorii care cuprind Age Range (interval de vârstă), Profanity (profanator), Heterosexual Themes (teme heterosexuale), Homosexual Themes (teme homosexuale), Nudity (nuditate), Violence (violenţă), Sex, Violence and Profanity (violenţă şi caracter profanator), Intolerance (intoleranţă), Glorifying Drug Use (glorificarea utilizării de droguri), Other Adult Themes (alte teme pentru adulţi) şi Gambling (jocuri de noroc). Din nou, interpretarea utilizatorului asupra înţelesului acestor categorii este slab definită. „Drug Use" include tutunul? Alcoolul? Cofeina? Insulina? Ceea ce pentru unii este inofensiv poate să pară de prost gust sau chiar imoral pentru alţii, în funcţie de părerile personale sau concepţiile religioase. Ţineţi cont că toate „drogurile" de mai sus sunt interzise de dogmele unei religii sau ale alteia, şi există mulţi părinţi în lume care pot fi deranjaţi de un sit care „glorifică" consumul de Pepsi-Cola sau de ceai cu gheaţă.

Explorarea sintaxei RDF

RDF arată aproape la fel cu orice altă aplicaţie XML. Trucul RDF este că veţi utiliza de obicei mai multe zone de nume pentru a permite chiar coexistenţa cadrului RDF cu descrierea efectivă a resursei. În cele mai multe cazuri, singura sintaxă RDF pe care trebuie să o cunoaşteţi este următoarea:

```
<rdf:Description about="URI">
       chestii extrase din alte zone de nume XML
</rdf:Description>.
```

În această sintaxă se foloseşte atributul `about` pentru a indica un URI care descrie resursa iar apoi puteţi adăuga orice doriţi pentru a uşura utilizarea sau descrierea resursei. Unele situri asigură conversia metadatelor Dublin Core direct în RDF sau în alte formate de metadate, un astfel de sit este de exemplu cel al U. K. Office for Library and Information Networking cu adresa `http://www.ukoln.ac.uk/metadata/dcdot/`.

Echivalentul RDF pentru valorile Dublin Core

În cele ce urmează puteţi vedea valorile RDF echivalente pentru valorile Dublin Core, aşa cum au fost generate automat de acest sit:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.0/">
  <rdf:Description about="http://www.leeanne.com/index.html">
    <dc:title>White Rose - LeeAnne.com</dc:title>
    <dc:creator>Lee Anne Phillips</dc:creator>
    <dc:subject>
      Lee Anne Phillips; consultant; author; telephony;
      networking; project manager
    </dc:subject>
    <dc:description>
      Home page of Lee Anne Phillips, LeeAnne.com, and White Rose
    </dc:description>
    <dc:publisher>White Rose</dc:publisher>
    <dc:date>26 Oct 1999</dc:date>
    <dc:type>Text.Homepage.Organizational</dc:type>
    <dc:format>text/html</dc:format>
    <dc:format>6000 bytes</dc:format>
    <dc:language>en-US</dc:language>
    <dc:coverage>global</dc:coverage>
    <dc:rights>Copyright (c) 1999 Lee Anne Phillips</dc:rights>
  </rdf:Description>
</rdf:RDF>
```

Observaţi că este de fapt acelaşi pachet de informaţii, dar elementele etichetei `<meta>` au fost convertite în elemente XML şi atât zona de nume RDF cât şi cea Dublin Core sunt referite, astfel încât singura etichetă RDF importantă, `Description`, să poată fi referită fără confuzii. Versiunea RDF este, din fericire, doar puţin mai vorbăreaţă decât versiunea nativă HTML, dar este oricum ceva mai voluminoasă.

Bineînţeles, RDF este ceva mai complicat decât am văzut în acest exemplu. Cuprinde şi un limbaj pentru scheme şi ceva sintaxă în plus pentru a vă permite să descrieţi relaţiile

dintre obiectele diferitelor zone de nume. Depinde de dumneavoastră dacă veți folosi sau nu această facilitate. Pur și simplu, nu este necesară în majoritatea cazurilor.

Sintaxa RDF prescurtată

Totuși, pentru documentele HTML este mai comodă varianta prescurtată a sintaxei, care vă permite să ascundeți elementele străine de analizorul HTML. Pur și simplu tratați toate elementele ca și cum ar fi atribute și le înghesuiți pe toate într-o etichetă Description:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.0/">
  <rdf:Description about="http://www.leeanne.com/index.html"
    dc:title="White Rose - LeeAnne.com"
    dc:creator="Lee Anne Phillips"
    dc:subject="Lee Anne Phillips; author; consultant; telephony;
              ➥networking; project manager"
    dc:description="Home page of Lee Anne Phillips, LeeAnne.com, and White Rose"
    dc:publisher="White Rose"
    dc:date="26 Oct 1999"
    dc:type="Text.Homepage.Organizational"
    dc:format="text/html"
    dc:format="6000 bytes"
    dc:language="en-US"
    dc:coverage="global"
    dc:rights="Copyright (c) 1999 Lee Anne Phillips" />
</rdf:RDF>
```

Sugestie de la LeeAnne.com Words to weave by

Prescurtările RDF economisesc o grămadă de timp și rezolvă multe probleme legate de interacțiunea dintre XML/RDF și browserele mai vechi.

Această versiune este puțin mai scurtă și are marele avantaj că textul nu va fi afișat de un browser HTML standard, deoarece acesta ar trebui să ignore etichetele pe care nu le înțelege. De fapt, browserul va vedea câteva spații în plus, pe care ar trebui să le ignore conform specificațiilor HTML.

Combinarea diferitelor scheme de metadate

Puteți combina într-o înregistrare diferite scheme de metadate, astfel încât nu sunteți limitat la utilizarea uneia sau a alteia. În exemplul următor, unele informații sunt adăugate pentru Global Information Locator Service (GILS – serviciul global pentru localizarea informațiilor). Sintaxa este exact aceeași, însă trebuie adăugată o nouă zonă de nume:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:gils="http://www.gils.net/"
  xmlns:dc="http://purl.org/dc/elements/1.0/">
  <rdf:Description about="http://www.leeanne.com/index.html"
    gils:Originator="Lee Anne Phillips"
    gils:Abstract="Home page of Lee Anne Phillips, LeeAnne.com, and White Rose"
    dc:title="White Rose - LeeAnne.com"
    dc:creator="Lee Anne Phillips"
    dc:subject="Lee Anne Phillips; author; consultant; telephony; networking;
              ➥project manager"
    dc:description="Home page of Lee Anne Phillips, LeeAnne.com, and White Rose"
    dc:publisher="White Rose"
    dc:date="26 Oct 1999"
    dc:type="Text.Homepage.Organizational"
    dc:format="text/html"
    dc:format="6000 bytes"
    dc:language="en-US"
    dc:coverage="global"
    dc:rights="Copyright (c) 1999 Lee Anne Phillips" />
</rdf:RDF>
```

Puteți astfel adăuga în continuare zone de nume și descrieri de resurse, atât cât este necesar.

Utilizarea resurselor RDF

RDF poate descrie orice obiect abstract cu ajutorul unui URI, chiar dacă obiectul nu este accesibil prin Web sau chiar dacă acesta nu există fizic. O carte, de exemplu, are un identificator ISBN unic, iar acesta poate funcționa atât ca identificator, cât și ca zonă de nume, însă acest număr este atribuit cu mult înainte de apariția cărții. Analog, puteți atribui un identificator unic Coloanelor lui Hercule sau Grădinilor suspendate din Babilon, deși acestea au dispărut cu milenii în urmă. Acestea sunt resurse, chiar dacă nu le puteți vizita și nu puteți face altceva decât să vi le imaginați. În general, orice lucru pe care îl puteți arăta, chiar numai imaginar sau metaforic, este o resursă. Predicatele (adică proprietățile) nu sunt resurse, deoarece nu puteți arăta „alergare" în absența unei persoane care face așa ceva.

Notă Resursele care nu sunt descrise printr-un URI sunt Uniform Resource Names (URN) în mod prestabilit. Ideal ar fi ca un URN să fie atribuit de o autoritate recunoscută, pentru a garanta unicitatea. Un URN are un prefix URN, de exemplu URN:*OriceAutoritate.OriceNumeAtribuit*.

În momentul de față, foarte puține obiecte abstracte de pe Web au metadate asociate. Autorii specificațiilor RDF speră că situația se va schimba cu timpul, pe măsură ce persoanele interesate în anumite domenii se vor convinge de utilitatea unor astfel de scheme, iar instrumentele de îmbinare a diferitelor scheme se vor îmbunătăți.

Resursele RDF sunt concepute ca obiecte, într-un sens „orientat spre obiecte". Descrierea limbajului este proiectată pentru a asigura moștenirea, chiar moștenirea multiplă, oferind posibilitatea ca un browser inteligent să parcurgă arborii familiilor de moștenitori pentru a afla cum trebuie procesată o schemă pe care nu a mai întâlnit-o.

Utilizarea proprietăților RDF

Ați văzut pe scurt cum se utilizează atributul about pentru a indica resursa descrisă de o pereche proprietate/valoare, făcând necesar al treilea termen al tripletului pentru o formulare RDF. În termeni gramaticali, resursa este complementul propoziției RDF. Proprietatea este predicatul, iar șirul de caractere este, în toate cazurile, subiectul. Subiectul poate fi o altă resursă, iar în acest caz acesta va avea asociată o altă pereche proprietate/valoare într-o altă propoziție, fostul subiect fiind acum complementul propo-

ziției. Veți vedea că acești termeni sunt folosiți în documentația XML când cu un sens când cu altul, deci ar trebui să vă obișnuiți cu ei.

În exemplul următor, membrii tripletului RDF sunt definiți prin denumiri sugestive:

```
<rdf:Description about="resourceURI">
       <property>literal</property>
</rdf:Description>
```

ceea ce poate fi transformat într-o propoziție: `resourceURI` are o proprietate (`PROPERTY`) care are valoarea `literal`.

Totuși, tripletul simplu complement, predicat și subiect nu este de ajuns în lumea reală. Uneori mai multe complemente au aceeași legătură cu un subiect. Fiecare propoziție trebuie duplicată pentru a păstra unitatea tripletelor? RDF permite o notație prescurtată pentru a descrie acest tip de relație multiplă. Deci dacă doriți să spuneți că Lee Anne a creat toate paginile acestei cărți, puteți folosi următoarea sintaxă care descrie o relație una-la-mai multe:

```
<lap:Book rdf:about="URN:ISBN-0789719967">
  <rdf:Seq ID="pages">
    <rdf:li resource="page1.doc" />
    <rdf:li resource="page2.doc" />
    <rdf:li resource="page3.doc" />
    <rdf:li resource="and so on …" />
  </rdf:Seq>
  <rdf:Description aboutEach="#pages">
    <DC:Creator>Lee Anne</DC:Creator>
  </rdf:Description>
</lap:Book>
```

Termenul RDF `aboutEach` distribuie descrierea dată pentru containerul secvențial RDF. Dacă ați fi folosit `about` în loc de `aboutEach`, propoziția ar fi spus doar că Lee Anne a creat secvența. Este foarte adevărat, dar nu este suficient. Observați și faptul că un URI nu trebuie neapărat să indice un loc anume pe Web. În acest exemplu indicați un obiect din lumea reală, chiar dacă nu există încă (cel puțin în momentul scrierii cărții) și are o referință metaforică, nu o cale urmată de o locație. Referința `aboutEach` este distributivă peste elementele unui container și *numai* peste elementele unui container. Nu poate fi folosită în nici un alt mod.

Mai există încă o referință distributivă care poate fi folosită pentru a asocia o pereche proprietate/valoare cu mai multe obiecte. Pentru asocierea unei mențiuni de copyright cu fiecare pagină dintr-un sit Web sau cu un director, puteți folosi următorul exemplu:

```
<lap:Sites>
  <rdf:Seq ID="pages">
    <rdf:li resource="http://www.leeanne.org/page1.html" />
    <rdf:li resource="http://www.leeanne.com/page1.html" />
    <rdf:li resource="http://www.leeanne.com/page2.html" />
    <rdf:li resource="http://www.leeanne.com/page3.html" />
    <rdf:li resource="and so on …" />
  </rdf:Seq>
  <rdf:Description aboutEachPrefix="http://www.leeanne.com/">
    <DC:Rights>Copyright [cp] 1999 Lee Anne Phillips</DC:Rights>
  </rdf:Description>
</lap:Sites>
```


Prima locație Web nu începe cu același prefix ca și celelalte, deci nu va primi proprietatea respectivă. Este foarte corect, din moment ce nu există încă. Toate celelalte sunt asociate cu declarația de copyright, exact așa cum doream.

Utilizarea valorilor RDF

Există două cazuri particulare de valori în RDF care pot fi utile. În primul, conținutul este declarat ca un `literal` RDF/XML, deci procesorul RDF va ignora orice etichetă inclusă în el. Se poate folosi, de exemplu, pentru a ascunde codul XHTML de procesorul RDF:

```
<lap:AddressLabel rdf:parseType="Literal">
  <xhtml:ul>
    <xhtml:li>Bill Clinton</xhtml:li>
    <xhtml:li>The White House</xhtml:li>
    <xhtml:li>Washington, DC</xhtml:li>
  </xhtml:ul>
</lap:AddressLabel>
```

În al doilea caz, poate trebuie să vă referiți la o entitate definită în altă parte. De exemplu, în cele ce urmează, Luna este descrisă ca fiind alcătuită din atâtea Gigatone de polistiren expandat, în care valoarea `Gigatone` este definită de National Institute of Standards and Technology – institutul național pentru standarde și tehnologie:

```
<RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:nist="http://www.nist.gov/units/">
    <rdf:Description about="URN:Moon">
      <nist:weight rdf:parseType="Resource">
        <rdf:material>styrofoam</rdf:material>
        <rdf:value>14522204300</rdf:value>
        <nist:units rdf:resource="http://www.nist.gov/units/Gigatone"/>
      </nist:weight>
    </rdf:Description>
</RDF>
```

Referi ițele, valorile și zonele de nume sunt aproape toate imaginare, deci nu vă așteptați să le găsiți undeva în situl National Institute of Standards and Technology.

Colecţii RDF

RDF cuprinde trei modele de containere – `bag`, `sequence` și `alternative` – care corespund în mare cu conținutul XML `any`, `sequence` și `choice`. Observați că `bag` nu este un set, deoarece permite explicit existența duplicatelor și descrie de fapt o colecție neordonată de resurse.

Prin utilizarea unui `bag` puteți defini un triumvirat fără să specificați nici un fel de ordine sau rang. De exemplu, să presupunem că ați definit o zonă de nume pentru istoria antică cu prefixul `AH` și doriți să vă referiți la Triumviratul din Roma antică fără să sugerați că unul dintre cei care conduceau avea mai multă putere și mai mult prestigiu decât ceilalți. Iată cum puteți face acest lucru:

```
<AH:Triumvirate>
  <rdf:Bag>
    <rdf:li>Crassus</rdf:li>
```

```
    <rdf:li>Pompeius</rdf:li>
    <rdf:li>Caesar</rdf:li>
  </rdf:Bag>
</AH:Triumvirate>
```

Dacă doriţi să arătaţi că *exista* o ordine, va trebui să folosiţi o secvenţă:

```
<AH:Triumvirate>
  <rdf:Seq>
    <rdf:li>Caesar</rdf:li>
    <rdf:li>Crassus</rdf:li>
    <rdf:li>Pompeius</rdf:li>
  </rdf:Seq>
</AH:Triumvirate>
```

Dacă însă doriţi să indicaţi o opţiune exclusivă între mai multe alternative, ceea ce s-a şi întâmplat în acest exemplu, veţi folosi o selecţie:

```
<AH:Triumvirate>
  <rdf:Alt>
    <rdf:li>Caesar</rdf:li>
    <rdf:li>Crassus</rdf:li>
    <rdf:li>Pompeius</rdf:li>
  </rdf:Alt>
</AH:Triumvirate>
```

Şi, în sfârşit, dacă doriţi să ascundeţi conţinutul colecţiilor dumneavoastră faţă de alte motoare, puteţi folosi sintaxa prescurtată pe care am văzut-o mai devreme, asociată cu o proprietate ordinală specială care este definită în Recomandare. Trebuie să procedaţi în acest fel deoarece XML nu vă permite să aveţi două atribute cu aceeaşi valoare într-o etichetă.

```
<AH:Triumvirate>
  <rdf:Alt
    rdf:_1="Caesar"
    rdf:_2="Crassus"
    rdf:_3="Pompeius" />
</AH:Triumvirate>
```

Aceste relaţii pot fi combinate în multe moduri, dar discuţia ar ocupa mai mult spaţiu decât ar merita. Recomandarea, pe care o puteţi găsi la `http://www.w3.org/TR/REC-rdf-syntax/`, este surprinzător de lizibilă, şi tot aici puteţi găsi şi Schema propusă de Recomandare, care ar trebui să fie deja o Recomandare în momentul în care citiţi aceste rânduri.

Când trebuie adăugat RDF la XML

Una dintre utilizările avute în vedere pentru RDF este adăugarea unor nume de elemente mai universale la numele de elemente XML. Această utilizare teoretică este controversată în momentul actual. Deşi prezintă avantaje pentru motoarele de căutare, prin faptul că, de exemplu, un utilizator danez poate să aibă o descriere RDF în engleză care să împacheteze numele de elemente daneze, pare destul de nesigur în practică.

Să presupunem că aţi definit în daneză o scurtă înregistrare:

```
<Titel>White Rose</Titel>
<Forfatter>Lee Anne Phillips</Forfatter>
```


Merită să suplimentaţi această informaţie cu:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.0/">
    <rdf:Description about="http://www.leeanne.com/">
      <dc:Title>White Rose</dc:Title>
      <dc:Creator>Lee Anne Phillips</dc:Creator>
    </rdf:Description>
</rdf:RDF>
```

sau cu *orice alt* vocabular uniform? De ce să nu utilizaţi de la început vocabularul uniform? Deşi poate că un vorbitor danez ar fi deranjat să folosească etichete în engleză, majoritatea danezilor cunosc oricum engleza, ca de altfel şi germana, şi poate chiar franceza. Deci nu e chiar aşa mare lucru, nu? Nu este aşa! Deoarece aceste lucruri pot fi definite ca şabloane XSLT şi pot fi generate automat după aceea, nu va trebui să vedeţi niciodată vocabularul standard în care se vor transforma etichetele dumneavoastră. Deci este la fel de simplu să codaţi în limba dumneavoastră şi să lăsaţi calculatorul să facă ce e mai greu.

Pe de altă parte, ar fi la fel de uşor să se traducă etichetele în acel vocabular standard, aşa că de ce să ne mai încurcăm cu RDF? Un motiv ar fi acela că utilizatorii naivi care se întâmplă să vorbească daneza pot să încerce căutări cu etichete care au sens în context. Desigur, ignorăm faptul că un danez ar putea la fel de uşor să se gândească la *Autor* ca la *Forfatter*. Dar, cel puţin, căutarea se restrânge şi oricum se poate căuta de două ori. Sinonime de acest fel există în multe limbi.

Sugestie de la LeeAnne.com Words to weave by

Deşi puteţi aştepta apariţia unor noi instrumente, acestea nu vor apărea înainte de a exista pe Web o anumită masă critică de date marcate. Gândiţi-vă de pe acum la adăugarea de metadate la toate paginile dumneavoastră şi astfel veţi contribui la accelerarea apariţiei instrumentelor.

Pare plauzibil că în cele din urmă browserele vor conţine o anumită formă de tehnologie încorporată, care va realiza automat toate aceste traduceri magice. Astfel, presupusul danez va putea căuta un *Forfatter* numit Lee Anne Phillips iar acest cuvânt va fi identificat fără greutate ca fiind din clasa de lucruri numite *Creator* în universul Dublin Core şi va fi căutat şi sub acest nume şi alte câteva pe deasupra.

Deşi în acest capitol am tratat lucrurile numai la suprafaţă, ştiţi cum să realizaţi 99% din operaţiile pe care într-adevăr doriţi să le faceţi, spre deosebire de alte cărţi, care vă spun totul, dar vă lasă să vă zbateţi pentru ceva concret.

În capitolul următor vom examina mai detaliat XHTML, deoarece acesta este limbajul bazat pe XML pe care îl vor întâlni majoritatea utilizatorilor în viitorul apropiat.

Depanarea RDF

RDF-ul meu funcţionează?

V-aţi creat resursa Web, aţi evaluat-o prin RSACi, aţi descris-o în metadate Dublin Core, aţi flirtat cu formele de metadate speciale ale AltaVista, iar acum vă întrebaţi dacă într-adevăr se întâmplă ceva.

Metadatele, prin natura lor, sunt alunecoase. Dacă funcționează corect, efectele nu sunt vizibile. În afară de cazul în care aveți acces la propriul dumneavoastră motor de căutare sau la alte mijloace prin care să descoperiți dacă RDF-ul dumneavoastră face ceea ce doreați, nu prea aveți ce face. Puteți deschide pagina într-un browser sau editor care validează și să verificați corectitudinea formulării, ceea ce este un exercițiu util pentru orice pagină XML, dar testul de bază este să încercați pagina. Dacă folosiți Internet Explorer, puteți evalua o pagină Web de test ca Adults Only și să vedeți dacă setările din Content Advisor acționează sau nu, adică dacă puteți accesa sau nu pagina. Pentru toate celelalte browsere, e mai greu de spus.

În afară de cazul în care aveți o aplicație care folosește metadate, nu se va întâmpla nimic atunci când dați peste ele. Acestea sunt extrem de greu de depanat. Depanarea constă în încărcarea paginilor în aplicația dumneavoastră, oricare ar fi aceasta, și compararea rezultatelor obținute cu cele așteptate.

XHTML în detaliu

CAPITOLUL 22

În acest capitol

- Utilizarea XHTML într-o lume HTML
- Minimizările HTML
- Corectarea particularităților HTML
- Extensibilitatea XHTML
- Transformarea HTML în XHTML
- Personalizarea XHTML
- Depanarea XHTML

Utilizarea XHTML într-o lume HTML

Pe scurt, XHTML este o reformulare a HTML ca aplicație a XML. Însă această afirmație simplă ignoră realitatea HTML în interpretarea actuală, conform căreia majoritatea browserelor comerciale ignoră multe erori flagrante de codare pentru a putea să afișeze ceva rezonabil, în ciuda sintaxei greșite. Afirmația ignoră de asemenea faptul că HTML este de fapt o aplicație a SGML și, așa cum am văzut mai devreme, XML *nu* este SGML. Printre lucrurile la care s-a renunțat prin trecerea de la SGML la XML este și capacitatea de a exclude elemente din definiția altor elemente. De asemenea, a fost eliminată capacitatea SGML de a abrevia construcții atunci când analizorul poate determina porțiunile de cod absente în timp ce examinează fluxul de date de intrare. Din acest motiv, puteți să uitați etichetele de încheiere pentru elementele paragraf <P> (și multe altele) în HTML, dar nu și în XHTML.

Minimizările HTML

Următorul fragment de cod este un document HTML complet și este legal în HTML, deși nu vă recomand acest tip de codare deoarece depinde într-o măsură prea mare de minimizările în stil SGML. Deși este legal în HTML, nu este deloc clar și este predispus la erori din neatenție:

```
<TITLE>HTML Example</TITLE>
<H1>HTML Example</H1>
<P>Paragraph One
<P>Paragraph Two
<BR>
```

```
<Table>
  <CAPTION>HTML Table</CAPTION>
  <TR>
    <TD>Element A1
    <TD>Element A2
    <TD>Element A3
  <TR>
    <TD>Element B1
    <TD>Element B2
    <TD>Element B3
</TABLE>
```

Observaţi că nu există eticheta <HTML>. Elementele HEAD şi BODY lipsesc deoarece un procesor HTML adevărat va deduce existenţa acestora din elementele TITLE şi H1 şi va insera elemente echivalente virtuale. De fapt, eticheta iniţială <P> nu este necesară, deoarece ar fi fost dedusă din existenţa textului din cod. De asemenea, etichetele de încheiere ale elementelor din tabel lipsesc, deoarece pot fi deduse uşor din gramatica HTML. De fapt, în mod normal în HTML, codarea etichetelor de încheiere pentru elementele unui tabel se omite uşor, deoarece acestea încarcă sursa cu etichete inutile şi îngreunează înţelegerea structurii documentului. În sfârşit, observaţi şi eticheta <Table> scrisă cu litere mari şi mici, care nu se potriveşte cu propria etichetă de încheiere, lucru perfect corect în HTML, dar în XML şi XHTML este perceput ca o eroare.

Translatarea acestui exemplu scurt în XHTML folosind programul Tidy al lui Dave Raggett are ca rezultat următorul exemplu XHTML construit corect:

Sugestie de la LeeAnne.com Words to weave by

Programul Tidy este oferit de W3C în forma de linie de comandă, la adresa http://www.w3.org/People/Raggett/tidy/; în schimb, pentru utilizatorii Windows, o împachetare GUI excelentă a fost realizată de cei de la Chami, http://www.chami.com/, şi este combinată cu un excelent editor HTML/XHTML, şi toate gratuit! Programul a fost portat pentru multe alte sisteme de operare, iar lista completă a acestor variante este păstrată pe situl lui Dave Raggett de la W3C.

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>XHTML Example</title>
  </head>
  <body>
    <h1>XHTML Example</h1>
    <p>
      Paragraph One
    </p>
    <p>
      Paragraph Two
    </p>
    <br />
    <table>
      <caption>XHTML Table</caption>
      <tr>
        <td>Element A1</td>
          <td>Element A2</td>
            <td>Element A3</td>
      </tr>
      <tr>
        <td>Element B1</td>
          <td>Element B2</td>
            <td>Element B3</td>
      </tr>
    </table>
  </body>
</html>
```

Observaţi că ieşirea Tidy a fost modificată puţin pentru a economisi spaţiu şi pentru a se conforma cu un stil de codare care uşurează înţelegerea tabelelor complexe. Elementele de tabel care se aliniază vertical când se afişează pagina sunt indentate manual pentru a alinia codul. Deşi este un exces pentru micul tabel dat ca exemplu, tabelele mai complexe devin adesea atât de greoaie în manipulare, încât modificarea ulterioară a acestora este o operaţie neplăcută şi predispusă la erori.

Sugestie de la LeeAnne.com Words to weave by

În cazul paginilor generate automat, nu trebuie să vă ocupaţi de frumuseţea formatării tabelelor, însă un număr surprinzător de mare de pagini sunt parţial modificate sau scrise în întregime de mână, chiar şi în siturile mari. Adăugarea unor indicii de formatare la cod vă va ajuta să vă daţi seama ce reprezintă fiecare element când, peste şase luni, vă veţi uita la pagină.

De asemenea, Tidy separă elementele antetului <h1> şi plasează conţinutul acestuia pe o linie nouă, rezultatul fiind, pentru multe browsere, o comportare diferită faţă de cea a unui element <h1> care are atât conţinutul, cât şi eticheta de încheiere pe aceeaşi linie. Acest comportament este un bug enervant al browserelor care afişează fişierul, şi nu al interpretării date de Tidy codului HTML. Din fericire, opţional, Tidy vă permite să eliminaţi complet indentările de conţinut dacă nu doriţi acest comportament. Însă eliminarea completă a indentărilor de conţinut are problemele sale; aşa încât, în acest caz, în loc să elimin complet indentările de conţinut, prefer să organizez manual ieşirea Tidy.

Tidy nu este unicul program de înfrumuseţare disponibil care să aibă ca scop doar verificarea corectitudinii codului HTML şi reformatarea sursei într-o formă mai structurată. Media++, o firmă germană (http://mpp.at/pretty/), oferă Pretty HTML, care este un program de înfrumuseţare fără o conexiune cu XML. Pretty HTML dispune de mai multe caracteristici care manipulează etichetele ASP şi celelalte adăugiri la standardul HTML, deci dacă platforma dumneavoastră ţintă este HTML sau Microsoft XML, atunci este posibil să îl preferaţi.

Am înlocuit, de asemenea, conţinutul care afişează „HTML“ cu „XHTML“ pentru a deosebi într-o oarecare măsură cele două coduri.

Tidy a realizat câteva modificări cu care nu vă puteţi juca. În primul rând, identificatorul XHTML şi elementele doctype au fost adăugate, iar zona de nume XHTML a fost identificată, aşa cum cer specificaţiile XHTML. În al doilea rând, toate etichetele sunt acum scrise cu litere mici, de asemenea o cerinţă a specificaţiilor XHTML. În al treilea rând, fiecare etichetă are o etichetă de încheiere corespunzătoare, deoarece XHTML cere ca

XML să fie construit corect. Elementele de încadrare <html>, <head> şi <body>, care pot fi omise în mod legal în HTML, nu sunt opţionale în XHTML. Acestea trebuie să facă parte dintr-un document XHTML valid, deci Tidy le adaugă în locul potrivit.

Elementele de paragraf sunt încheiate cu , elementele <tr> şi <td> sunt încheiate cu </tr> şi </td>, iar eticheta de linie goală <br> este încheiată cu un format special utilizat pentru elementele vide, <br />. Deşi XHTML vă permite de fapt să închideţi eticheta <br> cu eticheta de încheiere </br>, majoritatea browserelor existente nu înţeleg această sintaxă şi fac lucruri neaşteptate. Slash-ul de la sfârşitul etichetei de linie goală închide eticheta din punctul de vedere al XML şi XHTML, deşi majoritatea browserelor HTML actuale îl ignoră, deoarece ele „ştiu" că etichetele de linie goală sunt vide şi le închid în mod prestabilit.

Corectarea particularităţilor HTML

Următoarele particularităţi ale HTML trebuie corectate pentru ca un fişier HTML să fie considerat valid din punctul de vedere al XHTML:

- **Valorile atributelor** – Deşi SGML (şi, prin urmare, şi HTML) cere ca toate valorile text să fie închise între apostrofuri sau ghilimele în majoritatea cazurilor, au fost permise câteva excepţii pentru şirurile fără ambiguităţi. Deci, deşi un şir care conţine un spaţiu, cum ar fi "cauciuc rosu", trebuie întotdeauna să fie încadrat de ghilimele, în jurul unui şir fără echivoc, cum ar fi "minge", ghilimelele sunt opţionale. HTML permite , dar nu şi <p style= font-size: 12pt; >, deoarece nu are cum să ştie unde ar trebui să se încheie şirul. Un analizor sintactic SGML (sau HTML) are capacitatea de *anticipare*, adică de a prezice ceea ce va urma în fluxul de intrare, şi poate trata şirurile formate dintr-un singur cuvânt ca şi când ar fi închise între ghilimele, chiar dacă acestea sunt omise; totuşi, anticiparea nu poate să rezolve toate ambiguităţile. XML (şi XHTML) cere ca toate valorile pentru atribute să fie închise între ghilimele.
- **Atribute minimizate** – SGML (şi deci şi HTML) permite ca unele atribute să fie minimizate, deci un atribut care primeşte o singură valoare, dacă este prezent, poate fi indicat şi numai prin nume. Atributul ismap într-o etichetă <img>: de exemplu, <img src="URI" ismap="ismap"> poate fi indicat scriind simplu <img src="URI" ismap>; de fapt chiar aşa este definit în DTD-ul HTML. Această utilizare este o eroare în XHTML, deoarece acesta nu permite minimizările, deci întreaga valoare trebuie scrisă: <img src="URI" ismap="ismap">.
- **Etichetele de încheiere** – În XHTML toate elementele trebuie să fie închise, fie printr-o etichetă de încheiere obişnuită, fie prin sintaxa specială pentru etichete vide care pune încheierea într-o prescurtare XML a etichetei vide: <hr />. Deşi XML permite, teoretic, închiderea etichetelor vide exact la fel ca închiderea etichetelor cu conţinut, folosirea unei etichete de încheiere pentru un element vid blochează de obicei browserele HTML. Deci <hr></hr>, deşi este perfect valid, este de evitat.
- **Diferenţierea după tipul literelor** – HTML nu diferenţiază tipul literelor, însă XHTML face diferenţa. Atât numele elementelor XHTML cât şi numele atributelor trebuie să fie scrise cu litere mici. O excepţie apare în elementul XML <!DOCTYPE…> datorită definiţiei speciale a acestei declaraţii. XHTML a ales, pentru simplitate, să nu includă echivalente în majuscule pentru etichetele cu litere mici, în ciuda uriaşei baze instalate de documente HTML cu majuscule.

- **Script-uri şi foi de stil** – Deoarece script-urile şi foile de stil utilizează o sintaxă diferită de cea a HTML, browserele trebuie să ştie cum să le proceseze. Unele browsere nu se descurcă şi astfel s-a ajuns la practica destul de răspândită a încadrării script-urilor şi a conţinutului de stil în etichetele de comentarii SGML. XHTML poate să nu transmită deloc browserului informaţiile din comentarii, deci este mai bine să plasaţi atât script-urile cât şi foile de stil în fişiere externe, dacă este posibil. O alternativă este să plasaţi foaia de stil şi script-ul în interiorul unei secţiuni CDATA, aşa cum utilizatorii HTML pot plasa acelaşi conţinut în interiorul unui element comentariu. Totuşi, suportul pentru secţiunile CDATA este neregulat şi imprevizibil în browserele HTML, deci conţinutul poate să nu fie interpretat dacă este transmis browserului în acest mod. O soluţie pentru această problemă este discutată în secţiunea „Crearea unei pagini Web în XHTML" din acest capitol. Următorul exemplu de cod vă prezintă abordarea cu o secţiune CDATA, dar nu este recomandat să îl urmaţi:

```
<style>
  <![CDATA[
    Style sheet stuff…
  ]]>
</style>
```

- **Etichete opţionale** – HTML permite ca un număr surprinzător de mare de etichete să se deducă din context. În particular, chiar etichetele <html>, <head> şi <body> sunt toate opţionale, aşa cum aţi putut observa în exemplul din secţiunea de mai sus, „Minimizările HTML". Nici una dintre aceste etichete nu mai este opţională în XHTML.
- **Suportul lingvistic** – HTML foloseşte atributul lang pentru a indica limba folosită într-un document sau într-un element; XHTML foloseşte în acelaşi scop atributul xml:lang. Deoarece suportul pentru XML este neregulat şi este de presupus că va rămâne astfel, folosiţi în majoritatea situaţiilor ambele forme.
- **Suportul pentru seturi de caractere** – HTML aşteaptă o indicaţie referitoare la setul de caractere folosit într-un document, într-o etichetă meta din antetul documentului; XHTML aşteaptă aceeaşi informaţie în declaraţia XML. Deoarece suportul pentru XML este neregulat şi este de presupus că va rămâne astfel, folosiţi în majoritatea situaţiilor ambele forme. Deci, în antetul documentului <meta http-equiv='Content-type' content='text/HTML; charset="EUC-JP"' /> ar trebui să fie combinat cu <?XML version="1.0" encoding="EUC-JP" ?>.
- **Atribute speciale** – Din motive istorice HTML defineşte atributele name şi id cu înţelesuri mai mult sau mai puţin identice în diferite elemente, permiţând unui fragment URL să se refere la o anumită locaţie dintr-un document. În XHTML confuzia este eliminată într-o oarecare măsură deoarece atributul id este necesar în toate situaţiile, iar atributul name mai este permis doar în elementul ancoră <a>. Din păcate, ca urmare a acestui fapt, browserele mai vechi nu vor mai referi corect locaţiile din documente dacă este utilizat numai atributul id. În concluzie, ar trebui să utilizaţi ambele atribute id şi name cu aceeaşi valoare în fiecare element ancoră folosit ca ţintă, astfel: . De asemenea, deoarece browserele mai vechi nu permit alte ţinte decât elementele ancoră, ar trebui să restrângeţi ţintele la acestea.
- **Excluderi de elemente** – Deşi HTML conţine numeroase excluderi conforme cu SGML care reglementează elementele ce pot fi conţinute în alte elemente, XHTML

nu mai conține decât câteva reguli de excludere, care sunt destul de slabe. Deși aceste reguli pot să împiedice autorul să facă unele greșeli elementare, nu le previn în mod absolut. Regulile de excludere trebuie să fie aplicate manual de autor, iar lista elementelor excluse ar trebui să facă parte din ghidul fiecărui autor:

- Elementul <a> nu poate să conțină alte elemente <a>.
- Elementul <form> nu poate să conțină alte elemente <form>.
- Elementul <label> nu poate să conțină alte elemente <label>.
- Elementul <pre> nu poate să conțină elemente <big>, <small>, <sub>, <sup>, <img> sau <object>.
- Elementul <button> nu poate să conțină elemente <button>, <fieldset>, <form>, <iframe>, <input>, <label>, <select>, <isindex> sau <textarea>.

Mecanismele de siguranţă HTML

O altă problemă este că nici unul dintre numeroasele mecanisme de siguranță ale HTML, încorporate în browserele importante, nu va mai funcționa cu un analizor XHTML, deoarece aceste măsuri de siguranță se referă la încălcările cele mai frecvente ale DTD-ului HTML. De exemplu:

- **Elemente imbricate incorect** – Deși atât HTML cât și XHTML cer ca etichetele să fie închise în ordinea inversă a deschiderii lor, majoritatea browserelor vă permit să închideți multe dintre etichete în orice ordine doriți. În particular, elementele in-line care influențează aspectul textului, cum ar fi , <font> și altele, sunt afișate chiar dacă sunt închise la întâmplare. Deci, secvența distorsionată text este în mod incorect tratată ca și când ar fi vorba de secvența corectă text. În XHTML este legală doar secvența a doua, construită corect.
- **Spațiile** – Multe browsere tratează incorect liniile goale și spațiile în plus dintre atribute. Ar trebui să evitați liniile goale și spațiile în plus între valorile atributelor, deși atât HTML cât și XHTML cer în specificațiile lor ca spațiile în plus și liniile goale să fie reunite într-un singur spațiu.
- **Instrucțiunile de procesare** – Deși declarația XML și instrucțiunile de procesare ar trebui să fie ignorate de toate browserele, deoarece caracterele de încadrare < și > ar trebui să pară o etichetă necunoscută, de fapt majoritatea browserelor se încurcă în sintaxa <?…?> și tratează eticheta ca și conținut ce trebuie afișat. Evitați, dacă este posibil, instrucțiunile de procesare, deși de o declarație XML veți avea nevoie dacă doriți să utilizați și seturi de caractere diferite de cele prestabilite.

Extensibilitatea XHTML

Ați mai aflat despre extensibilitate în Capitolul 4, „Extinderea definiției tipului documentului prin modificări locale", și ați văzut un exemplu exhaustiv în care a fost adăugat un singur element, deci nu mai trebuie să repetăm efortul și aici. Totuși, trebuie să rețineți că există multe modalități de extindere a unui document XHTML. Deoarece XHTML are propria sa zonă de nume, putem încorpora cod XML din alte zone de nume folosind sintaxa zonei de nume pe care am discutat-o în Capitolul 6, „Zone de nume XML". Putem de asemenea să includem documente externe folosind facilitățile de legare ale XML.

Problemele de imbricare ale XHTML

Despre problemele de imbricare ale XHTML ați aflat în discuția noastră de mai înainte, dar deoarece acestea sunt o povară apăsătoare pentru autor, merită să le repetăm aici:

- Elementul <a> nu poate să conțină alte elemente <a>.
- Elementul <form> nu poate să conțină alte elemente <form>.
- Elementul <label> nu poate să conțină alte elemente <label>.
- Elementul <pre> nu poate să conțină elemente <big>, <small>, <sub>, <sup>, <img> sau <object>.
- Elementul <button> nu poate să conțină elemente <button>, <fieldset>, <form>, <iframe>, <input>, <label>, <select>, <isindex> sau <textarea>.

DTD-ul XHTML se referă la aceste limitări direct în definițiile acestor elemente, dar nu extinde efectiv nivelul primar de excludere și la elementele copil. Deci, elementul <a> poate să conțină un element conform DTD-ului, iar acesta poate la rândul său să conțină un element <a>, de asemenea în conformitate cu DTD-ul, în ciuda faptului că elementului <a> nu i se „permite" să conțină un element <a>, printr-o anexă textuală a Recomandării. În concluzie, autorul trebuie să urmărească nivelul de imbricare al elementelor în timp ce scrie, deoarece un analizor care validează nu va intercepta astfel de greșeli, sau analizorul XHTML ar trebui să conțină informații secrete despre fișierele XHTML care nu sunt conținute în DTD, lucru care încalcă chiar principiile de bază ale XML.

Deci, următorul fragment de cod XHTML este ilegal:

```
<a href="URI-1"><em>Exemplu <a href="URI-2">element</a> imbricat</em></a>
```

Însă același cod este „valid" după DTD-ul XHTML. Este interzis numai în textul anexei recomandării, nu în DTD-ul propriu-zis. Și următorul fragment este ilegal:

```
<a href="URI-1">Exemplu <a href="URI-2">element</a> imbricat</a>
```

Deoarece elementul nu este imbricat, greșeala va fi interceptată de orice analizor care validează și va determina o eroare fatală. Deși ambele exemple sunt erori, numai cea din urmă va fi interceptată de un analizor care validează.

Preprocesarea specială a XHTML

Singura soluție evidentă pentru această problemă este preprocesarea specială, atunci când se creează un număr mare de fișiere XHTML. Aceasta este o deficiență intenționată în proiectarea XML, deși evitarea sa *este posibilă*. Excluderile necesită o procesare cu anticipare, ducând la o complexitate enormă a analizorului care ar reuși să le rezolve. Deoarece complexitatea prea mare încalcă unul dintre principiile de design al XML, compromisul a fost considerat rezonabil. Cu timpul, prin exerciții, este posibil să țineți minte cele câteva reguli. Scrierea unui preprocesor cu anticipare nu este *deloc* complicată, deoarece problema este perfect înțeleasă. Deci complexitatea cerută poate fi deplasată într-un instrument care este folosit mai rar la crearea documentelor și exclusă din analizorul folosit la validarea după DTD. Pe de altă parte, comportarea unui browser, dacă o astfel de eroare trece neobservată, este nedefinită.

Sugestie de la LeeAnne.com Words to weave by

Având în vedere că interceptarea erorilor este un lucru bun, implementarea unui limbaj cum este XHTML, care permite erori ce nu pot fi detectate, prezintă o slăbiciune serioasă. Totuşi, detectarea acestor erori nu ar putea fi permisă în XML fără restricţii. Deci proiectanţii au făcut tot posibilul, printr-o excludere explicită în modelul de conţinut al unor elemente; totuşi, această excludere nu se extinde dincolo de contextul imediat al etichetei.

Morala este că nu trebuie să scăpaţi astfel de erori. O abordare pentru prevenirea erorilor este folosirea unui validator HTML sau XHTML, sau chiar a mai multor validatoare pentru verificarea codului. Codul din exemplul iniţial este un test bun, adăugând în el un element ancoră imbricat pentru ca validatoarele să poată găsi ceva incorect:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>XHTML Example</title>
  </head>
  <body>
    <h1>XHTML Example</h1>
    <p>
      Paragraph One
    </p>
    <p>
      Paragraph Two
    </p>
    <a name="myid" id="myid"><em>Outer Anchor
    <a name="idtwo" id="istwo">Inner Anchor</a></em></a>
    <br />
    <table>
      <caption>XHTML Table</caption>
      <tr>
        <td>Element A1</td>
          <td>Element A2</td>
            <td>Element A3</td>
      </tr>
      <tr>
        <td>Element B1</td>
          <td>Element B2</td>
            <td>Element B3</td>
      </tr>
    </table>
  </body>
</html>
```

Figura 22.1 vă prezintă rezultatul verificării codului XHTML modificat, prin intermediul setului de validare on-line WebLint, la `http://validator.w3.org/`. Deşi sunt prezentate mai multe erori, majoritatea nu par serioase deoarece se referă la codul tipic XML şi nu la sintaxa HTML. Totuşi, o eroare este descoperită exact aşa cum ne aşteptam. Elementul ancoră pe care l-am adăugat este indicat ca fiind imbricat în mod incorect, ceea ce este exact.

Validatorul SGML al W3C indică eroarea de imbricare şi, deoarece cunoaşte XHTML, nu mai indică şi alte erori. Observaţi că informaţiile speciale care nu fac parte din DTD, aşa cum am mai spus de câteva ori, sunt incluse în ambele validatoare. Acest fişier este deschis corect în toate analizoarele cu validare pe care le-am încercat, ceea ce este logic, dacă studiaţi DTD-ul.

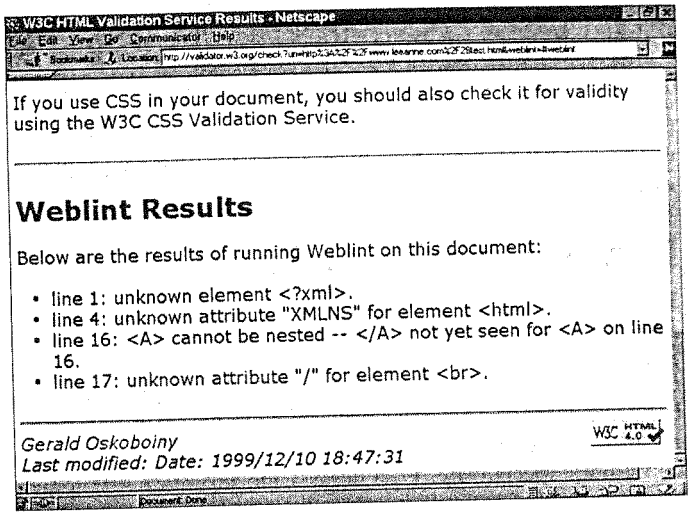


FIGURA 22.1
Rezultatele validării codului XHTML cu un validator HTML.

Transformarea HTML în XHTML

Transformarea HTML în XHTML este astfel proiectată încât să fie simplă. De fapt, pentru a evita unele din problemele de validare ale XHTML, poate este mai bine să creaţi mai întâi o pagină HTML validă folosind oricare dintre editoarele excelente care şi validează, majoritatea acestora fiind ieftine sau chiar gratuite. După ce testaţi aspectul paginii în mai multe browsere, la fel cum aţi proceda pentru o pagină HTML obişnuită, puteţi converti codul în XHTML. Conversia la XHTML vă asigură că numeroasele probleme (de imbricare sau altele) ale XHTML sunt interceptate de la început. De asemenea, pagina dumneavoastră XHTML va fi şi în format HTML valid.

Apoi, pentru a realiza efectiv transformarea, folosiţi un instrument de conversie, cum ar fi HTML-Kit de la Chami, deoarece aproape orice instrument automat este mult mai eficient şi mult mai puţin predispus la erori decât cea mai bună încercare de conversie manuală. HTML-Kit foloseşte Tidy, deci particularităţile Tidy trebuie avute în vedere şi corectate pentru a desăvârşi codul, dar chiar şi automatizarea acestei operaţii de curăţare este o sarcină uşor de realizat în orice editor de programare decent. Eu folosesc Visual SlickEdit (`http://www.slickedit.com/`), care dispune de aptitudini puternice pentru macroinstrucţiuni şi de un limbaj de programare complet, ce permite instalarea în editor a unor caracteristici de complexităţi diferite, a căror prezenţă pare chiar naturală.

Programul Tidy este disponibil şi în cod sursă, deci îl puteţi folosi ca bază pentru versiunea dumneavoastră personalizată şi să definiţi, dacă doriţi, propriile rutine de *tipărire sugestivă*.

Sugestie de la LeeAnne.com Words to weave by

Tipărirea sugestivă (pretty-printing), uneori numită şi *înfrumuseţare*, este un termen folosit în dezvoltarea de software şi se referă la formatarea automată a documentului de cod conform cu unele reguli de indentare şi de inserare a liniilor vide, pentru a mări lizibilitatea şi pentru a sublinia construcţiile imbricate ale limbajului.

Se poate lucra şi în sens invers, folosind un instrument de validare HTML pentru a valida paginile dumneavoastră XHTML. În acest fel vi se vor raporta în mod sigur câteva excepţii, poate chiar mai multe, dar ar trebui să fie destul de simplu să vă daţi seama care din mesajele de eroare sunt corecte şi care sunt rezultatul citirii porţiunilor de cod XHTML ca HTML incorect. În exemplul anterior, citirea fişierului XHTML într-un validator HTML ar trebui să genereze doar un mesaj de eroare referitor la atributul `xmlns` şi nimic altceva.

Personalizarea XHTML

Deşi am discutat un exemplu simplu de personalizare în Capitolul 4, în cele din urmă este vorba de mult mai mult de atât, deşi rămâne de văzut exact despre ce va fi vorba. Specificaţia XHTML este descompusă în module, fiecare pachet dispunând de subseturi complet funcţionale ale XHTML. Astfel se speră că se va putea modulariza întregul limbaj pentru folosirea în telefoane mobile sau în altfel de dispozitive, care se presupune că nu au nevoie de întreaga funcţionalitate a XHTML, sau pentru a adresa nevoilor unor comunităţi specializate. Deşi această modularizare este logică, nu este clar în ce mod aceste aptitudini modularizate vor adăuga ceva la ceea ce se poate realiza deja folosind zonele de nume.

Nici mecanismul de modularizare nu este încă foarte clar. Metoda prin care un DTD este descompus în subseturi modulare nu este definită şi pare să mărească considerabil complexitatea unui analizor care validează, deoarece acesta va trebui să realizeze o procesare condiţională a DTD-ului. Aceeaşi abordare, ca un meniu de pizza, folosită în SGML poate să fie utilă şi în XHTML, dar complexitatea generală a SGML sugerează că această operaţie nu va fi una simplă.

Profilul documentului XML

Recomandarea XHTML sugerează că o componentă numită profilul documentului va specifica facilităţile utilizate în procesarea documentelor de acel tip. Aceste facilităţi pot să cuprindă formate de imagini care pot fi folosite, nivelurile de suport pentru script-uri, suportul pentru foi de stil, subseturile modulare ale XHTML, facilităţile extinse pentru alte comunităţi de utilizatori şi aşa mai departe. Teoretic, profilul documentului va permite unui procesor XML să creeze dinamic DTD-uri particularizate sau alte tipuri de scheme de documente din mai multe părţi componente. Foarte probabil, folosind DTD-urile actuale, acest lucru va fi destul de dificil. Din nou, nu este clar în ce mod această facilitate va îmbunătăţi abordarea cu zone de nume, deoarece este de presupus că problemele rezolvate acum prin zone de nume va trebui să fie tratate într-un alt mod.

De asemenea, nu este clar care este avantajul acestei abordări faţă de combinaţia de DTD-uri specializate bine realizate, cum ar fi SMIL şi MathML, folosite împreună cu CSS sau XSL, care dublează multe din funcţiile de formatare a afişării ale XHTML.

În orice caz, problema suportului pentru browsere mai vechi probabil că nu va dispărea niciodată. Mai sunt încă persoane pe Web care folosesc browserul original Mosaic şi sunteţi extrem de optimist dacă vă aşteptaţi ca fiecare utilizator să treacă la cel mai nou browser. Acest lucru nu se va întâmpla decât în anumite comunităţi.

Crearea unei pagini Web XHTML

În acest exemplu, o pagină simplă este transformată din HTML în XHTML folosind HTML-Kit, dar păstrând compatibilitatea cu browserele existente în cel mai mare grad posibil. Deşi setările exacte ale HTML-Kit necesare pentru a realiza aceste operaţii nu fac obiectul acestui capitol, acestea sunt uşor de realizat folosind elementul de meniu Edit Preferences. Iată codul sursă HTML:

```
<HTML>
  <HEAD>
    <TITLE>Hypatia: A Notable Woman of Science</TITLE>
  <BODY>
    <DIV align="center"><IMG src="Hypatia-text-lc.gif"
         width="300" height="75" border="0"
         hspace="10" alt="[Hypatia's Name]" align="center"></DIV>
    <H1 align="center">Hypatia: A Notable Woman of Science</H1>
    <P>
    <DIV align="center"><IMG src="hypatia6.gif" width="280"
         height="340" border="0" hspace="10"
         alt="[Portrait of Hypatia]" align="center"></DIV>
    <P>
    <DIV align="center"><IMG src="Hypatia-text-uc.gif" width="300"
         height="75" border="0" hspace="10" alt="[Hypatia's Name]"
         align="center"></DIV>
    <A name=hercity></A>
    <P>Hypatia lived in the Egyptian city of Alexandria at the turn
    of the fourth century of the common era. Her father was the last
    director of the great Library of Alexandria and she had been able
    to study mathematics, astronomy, and philosophy to the extent
    that she was renowned as one of the greatest minds of her day,
    surpassing even the scholarship and reputation of her father.
    <A name=herdeath></A>
    <P>She herself was murdered in 415 C.E. at about 45 years of age
    and her work, which included scrolls on Diophantus's
    "Arithmetica," Apollonius's "Conics" and Ptolemy's astronomical
    theories, were systematically destroyed. Her legacy was passed
    down through her students and helped to form the basis of modern
    mathematics.
    <A name=herinventions></A>
    <P>She also invented early astronomical and mechanical devices
    such as the planar astrolabe and the hydroscope.
  </BODY>
</HTML>
```

Pentru a verifica acest cod folosind Tidy, fişierul este deschis în HTML-Kit sub Windows. De fapt nu verificăm efectiv codul, deoarece ştim deja (din cunoştinţele anterioare – trebuie să mă credeţi) că acest cod este valid în HTML (cu excepţia atributului `align=center` din etichetele `image`, care trebuie să fie inclus pentru compatibilitate cu ver-

siunile mai vechi ale multor browsere). În schimb, Tidy este folosit pentru transformarea codului în XHTML, adăugarea suportului pentru foi de stil și pentru tipărirea sugestivă.

După realizarea conversiei, selectați codul formatat din fereastra din dreapta, îl copiați peste codul original din fereastra din stânga, iar apoi salvați documentul. În acest moment, poate că veți dori să editați puțin codul. HTML-Kit este chiar un editor HTML/XHTML destul de bun și ușurează această operație. Figura 22.2 vă prezintă programul HTML-Kit la lucru.

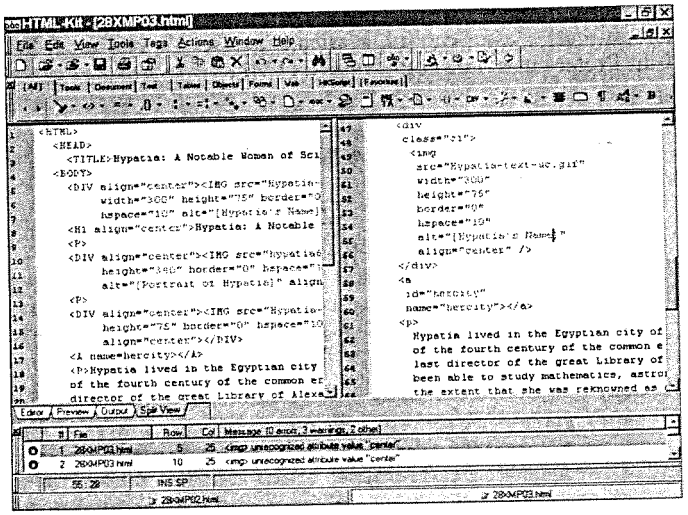


FIGURA 22.2
HTML-Kit vă este prezentat după transformarea și înfrumusețarea fișierului HTML Hypatia.

Următorul cod este editat puțin, dar din punct de vedere sintactic este reproducerea exactă a codului transformat:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Hypatia: A Notable Woman of Science</title>
<style type="text/css">
 h1.c2 {text-align: center}
 div.c1 {text-align: center}
</style>
  </head>
  <body>
    <div class="c1">
      <img src="Hypatia-text-lc.gif" width="300" height="75"
       border="0" hspace="10" alt="[Hypatia's Name in Greek]"
       align="center" />
    </div>
    <h1 class="c2">Hypatia: A Notable Woman of Science</h1>
    <p>
    </p>
    <div class="c1">
      <img src="hypatia6.gif" width="280" height="340" border="0"
       hspace="10" alt="[Portrait of Hypatia]" align="center" />
    </div>
    <p>
    </p>
    <div class="c1">
      <img src="Hypatia-text-uc.gif" width="300" height="75"
       border="0" hspace="10" alt="[Hypatia's Name in Greek]"
       align="center" />
    </div>
    <a id="hercity" name="hercity"></a>
    <p>
      Hypatia lived in the Egyptian city of Alexandria at the turn
      of the fourth century of the common era. Her father was the
      last director of the great Library of Alexandria and she had
      been able to study mathematics, astronomy, and philosophy to
      the extent that she was renowned as one of the greatest
      minds of her day, surpassing even the scholarship and
      reputation of her father.
    <a id="herdeath" name="herdeath"></a>
    </p>
    <p>
      She herself was murdered in 415 C.E. at about 45 years of age
      and her work, which included scrolls on Diophantus's
      "Arithmetica," Apollonius's "Conics" and Ptolemy's
      astronomical theories, were systematically destroyed. Her
      legacy was passed down through her students and helped to
      form the basis of modern mathematics.
    <a id="herinventions" name="herinventions"></a>
    </p>
    <p>
      She also invented early astronomical and mechanical devices
      such as the planar astrolabe and the hydroscope.
    </p>
  </body>
</html>
```

HTML-Kit și Tidy au realizat cu succes transformarea de la HTML la XHTML. Observați că au fost adăugate declarațiile XML și doctype, numele țintelor din etichetele ancoră au fost dublate automat, toate numele de elemente și atribute sunt scrise acum cu litere mici, valorile atributelor sunt acum puse între ghilimele și etichetele de încheiere au fost adăugate acolo unde lipseau.

Totuși, mai puteți face câteva lucruri, iar codul de mai sus a fost puțin modificat pentru a ocupa mai puțin loc în pagină. Însă nu am urmat unele din sugestiile pentru autori și ar trebui să explicăm câteva lucruri. Mai întâi, deși atributele nerecomandate `align=center` au fost înlocuite cu comenzi pentru foaia de stil în elementele `div`, au rămas neatinse în elementele `img` incluse. Este logic, deoarece multe browsere mai vechi nu recunosc elementele `div`, dar recunosc sintaxa `align="center"`, deci aceasta nu trebuie să fie modificată.

Pe de altă parte, foaia de stil este plasată în formă brută în document și va fi afișată ca

text în multe browsere. În mod ideal, ar trebui să încadrați comenzile de stil într-o secțiune CDATA ca aceasta:

```
<style type="text/css">
  <![CDATA[
    h1.c2 {text-align: center}
    div.c1 {text-align: center}
  ]]>
</style>
```

Totuși, acest fragment nu va fi interpretat de multe browsere și poate fi afișat de asemenea ca date, în funcție de ceea ce va presupune browserul despre sensul construcției respective. O variantă este să plasați informațiile de stil în interiorul comentariilor, așa cum se face de obicei în HTML, pentru a împiedica browserele mai vechi să interpreteze codul ca text ce trebuie afișat.

```
<style type="text/css">
  <!--
    h1.c2 {text-align: center}
    div.c1 {text-align: center}
 // -->
</style>
```

Dar și această metodă are propriile probleme. Conform Recomandării XML, analizoarele XML nu sunt obligate să transmită aplicației informațiile din comentarii, ceea ce înseamnă că informațiile de stil pot să dispară înainte de a fi văzute de browser, iar foaia de stil va fi goală atunci când browserul va încerca să o aplice documentului.

Deci soluția naivă pare să fie plasarea foii de stil într-un fișier extern, astfel:

```
<style type="text/css" src="hypatia.css"></style>
```

Din nefericire, multe browsere nu recunosc foile de stil externe. Deci unica opțiune rămasă, care pare să facă utilizabilă foaia de stil în majoritatea browserelor, este să plasați aceeași informație atât în foaia de stil externă, cât și în codul in-line inclus în comentarii:

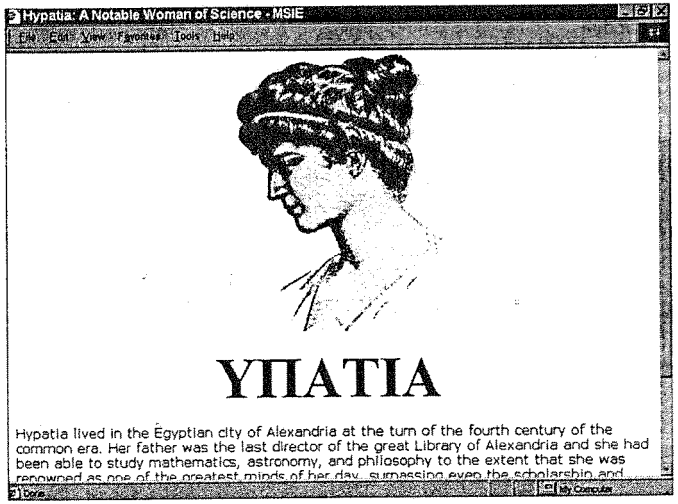
```
<style type="text/css">
  <!--
    h1.c2 {text-align: center}
    div.c1 {text-align: center}
 // -->
</style>
```

Browserele mai vechi care nu recunosc foile de stil externe vor folosi informațiile in-line. Browserele și mai vechi, cum ar fi Mosaic sau Netscape 2.0, care nu recunosc deloc foile de stil, vor ignora complet etichetele și datele. Deoarece se presupune că fișierul extern are oricum prioritate, dacă datele in-line dispar deoarece analizorul XML nu le-a transferat aplicației, nu e nici o pagubă. Foaia de stil va fi aplicată documentului de orice browser care este capabil să o interpreteze. Iată codul final:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Hypatia: A Notable Woman of Science</title>
<style type="text/css" src="hypatia.css">
  <!--
    h1.c2 {text-align: center}
    div.c1 {text-align: center}
 // -->
</style>
  </head>
  <body>
    <div class="c1">
      <img src="Hypatia-text-lc.gif" width="300" height="75"
       border="0" hspace="10" alt="[Hypatia's Name in Greek]"
       align="center" />
    </div>
    <h1 class="c2">Hypatia: A Notable Woman of Science</h1>
    <p>
    </p>
    <div class="c1">
      <img src="hypatia6.gif" width="280" height="340" border="0"
       hspace="10" alt="[Portrait of Hypatia]" align="center" />
    </div>
    <p>
    </p>
    <div class="c1">
      <img src="Hypatia-text-uc.gif" width="300" height="75"
       border="0" hspace="10" alt="[Hypatia's Name in Greek]"
       align="center" />
    </div>
    <a id="hercity" name="hercity"></a>
    <p>
      Hypatia lived in the Egyptian city of Alexandria at the turn
      of the fourth century of the common era. Her father was the
      last director of the great Library of Alexandria and she had
      been able to study mathematics, astronomy, and philosophy to
      the extent that she was renowned as one of the greatest
      minds of her day, surpassing even the scholarship and
      reputation of her father.
    <a id="herdeath" name="herdeath"></a>
    </p>
    <p>
      She herself was murdered in 415 C.E. at about 45 years of age
      and her work, which included scrolls on Diophantus's
      "Arithmetica," Apollonius's "Conics" and Ptolemy's
      astronomical theories, were systematically destroyed. Her
      legacy was passed down through her students and helped to
      form the basis of modern mathematics.
  <a id="herinventions" name="herinventions"></a>
    </p>
    <p>
      She also invented early astronomical and mechanical devices
      such as the planar astrolabe and the hydroscope.
    </p>
  </body>
</html>
```

Această variantă pare să fie cel mai bun compromis între accesibilitate și respectarea Recomandării XML. Deși mai sunt câteva atribute nerecomandate, în rest codul este curat. Figura 22.3 vă prezintă parțial pagina rezultată, folosind Microsoft Internet Explorer 5.0.

FIGURA 22.3
Microsoft Internet Explorer 5.0 vă înfățișează pagina Hypatia completă.



Totuși, munca nu este încă încheiată; acum trebuie să depanați rezultatul în alte browsere.

Într-un fel, disponibilitatea atâtor instrumente automate pentru transformarea HTML, care conține el însuși multe instrumente de validare, în XHTML ar trebui să ducă la o depanare banală a XHTML. Totuși, după cum ați văzut și în exemplul anterior, este evident că aveți de luat câteva decizii referitoare la cod, chiar dacă acesta va fi rezultatul unui instrument automat. Iar cel mai trist lucru este că browserele sunt creaturi nesigure, în care nu puteți avea încredere.

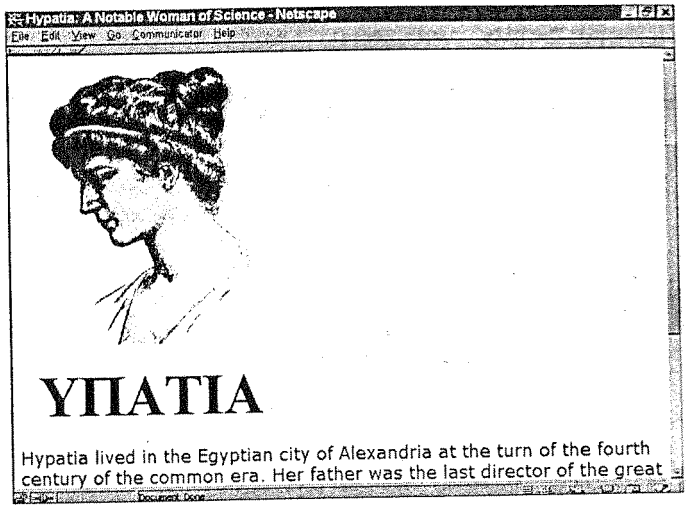
Depanarea XHTML

Paginile dumneavoastră sunt date peste cap de incompatibilităţile browserelor

După toată munca depusă, atunci când deschideți fișierul transformat în Netscape Navigator 4.7, dintr-un anumit motiv, alinierea centrală nu funcționează. Figura 22.4 vă prezintă rezultatul anormal.

Hmm. Cum este posibil? Nici una dintre versiunile foii de stil nu pare să influențeze pagina, cel puțin nu în mod vizibil. Verificați cu atenție codul, numele fișierului CSS de pe disc și orice vă mai trece prin cap. Acum studiați codul original, care funcționa corect în toate browserele (credeți-mă – așa este). Singura diferență pertinentă pare să fie atributul de aliniere care a fost scos din cod și adăugat în foaia de stil. Încercați-l. Funcționează! Ticăloșii de programatori de la Netscape au folosit un motor CSS care nu recunoaște și nu aplică în mod corect alinierea de text în document, deci trebuie să puneți atributul la locul său.

FIGURA 22.4
Netscape Navigator se pierde atunci când încearcă să afișeze o pagină „standard".



Iată codul rezultat:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Hypatia: A Notable Woman of Science</title>
<style type="text/css" src="hypatia.css">
  <!--
    h1.c2 {text-align: center}
    div.c1 {text-align: center}
 // -->
</style>
  </head>
  <body>
    <div class="c1" align="center">
      <img src="Hypatia-text-lc.gif" width="300" height="75"
       border="0" hspace="10" alt="[Hypatia's Name in Greek]"
       align="center" />
    </div>
    <h1 class="c2" align="center">Hypatia: A Notable Woman of Science</h1>
    <p>
    </p>
    <div class="c1" align="center">
      <img src="hypatia6.gif" width="280" height="340" border="0"
       hspace="10" alt="[Portrait of Hypatia]" align="center" />
    </div>
    <p>
    </p>
    <div class="c1" align="center">
```

```
    <img src="Hypatia-text-uc.gif" width="300" height="75"
     border="0" hspace="10" alt="[Hypatia's Name in Greek]"
     align="center" />
  </div>
  <a id="hercity" name="hercity"></a>
  <p>
    Hypatia lived in the Egyptian city of Alexandria at the turn
    of the fourth century of the common era. Her father was the
    last director of the great Library of Alexandria and she had
    been able to study mathematics, astronomy, and philosophy to
    the extent that she was renowned as one of the greatest
    minds of her day, surpassing even the scholarship and
    reputation of her father.
  <a id="herdeath" name="herdeath"></a>
  </p>
  <p>
    She herself was murdered in 415 C.E. at about 45 years of age
    and her work, which included scrolls on Diophantus's
    "Arithmetica," Apollonius's "Conics" and Ptolemy's
    astronomical theories, were systematically destroyed. Her
    legacy was passed down through her students and helped to
    form the basis of modern mathematics.
 <a id="herinventions" name="herinventions"></a>
  </p>
  <p>
    She also invented early astronomical and mechanical devices
    such as the planar astrolabe and the hydroscope.
  </p>
 </body>
</html>
```

Înapoi la bazele unei compatibilităţi complete

Deşi singurele modificări au fost adăugarea atributelor `align="center"` în cele două imagini şi în titlul de nivelul 1, pagina este acum afişată corect. În realitate pot să apară o mulţime de astfel de probleme, iar faza de depanare poate să dureze puţin mai mult. Figura 22.5 vă prezintă pagina depanată în Netscape Navigator 4.7.

Deşi specificaţiile sunt lucruri minunate, lumea reală a dezvoltării de cod cere compromisuri între modul în care ar trebui făcute lucrurile şi realităţile vieţii. În multe cazuri puteţi alege între a face totul „aşa cum trebuie" sau în maniera Microsoft, iar pentru câteva lucruri esenţiale ar trebui să acceptaţi şi unele compromisuri. Deoarece atât Netscape cât şi Microsoft sunt conştiente de propria lor istorie, valoarea `align="center"` din eticheta `<img>` nu va pune probleme în browserele actuale ale amblor firme.

În această secţiune aţi urmărit dezvoltarea unui document XHTML de la rădăcinile sale HTML, inclusiv compromisurile necesare între standarde şi realitatea browserelor, o caracteristică obişnuită în crearea documentelor pentru Web. Este posibilă adăugarea valorii atributului `<img align="center" ...>` în DTD-ul intern, dacă vi se pare necesar, dar face mai mult rău decât bine în momentul de faţă, deoarece includerea unei submulţimi interne a DTD-ului blochează multe browsere HTML.

BIBL. CENTR. UNIV. „M. EMINESCU" INFORMATICĂ

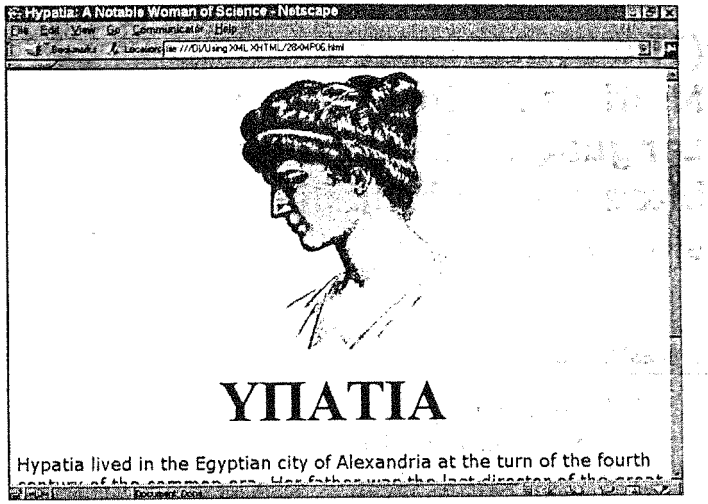


FIGURA 22.5
Acum, Netscape Navigator afişează corect pagina.

Utilizarea SMIL

CAPITOLUL

23

(The Synchronized Multimedia Integration Language – limbajul de integrare multimedia sincronizat)

În acest capitol

SMIL – ruda îndepărtată a lui HyTime
Explorarea sintaxei SMIL
Modelul temporal al SMIL
Depanarea SMIL

SMIL – ruda îndepărtată a lui HyTime

SMIL (*The Synchronized Multimedia Integration Language* – limbajul de integrare multimedia sincronizat) este în mod esențial o listă de editări și de legături la fișiere externe care conțin esența unei prezentări. SMIL seamănă cu notele de dans ale unui coregraf sau cu instrucțiunile de mișcare în scenă ale unui regizor de teatru, care permit actorilor să intre și să iasă din scenă la replica potrivită, dar care nu realizează și interpretarea în sine. Editarea fișierului este mult ușurată de faptul că toate metadatele și informațiile de control sunt conținute în documentul SMIL, iar nimic din conținutul multimedia efectiv nu este inclus în acesta.

Cea mai mare parte din funcționalitatea SMIL este dată de numai două elemente: elementul par (parallel – paralel) descrie lucruri care trebuie să se realizeze mai mult sau mai puțin simultan și elementul seq (sequential – secvențial) descrie lucruri care trebuie să se realizeze unul după celălalt. Lucrurile care trebuie realizate sunt referite prin omniprezentele URI, succesoarele definite de W3C pentru URL-uri. Ceea ce mai rămâne din SMIL este cadrul de lucru și detaliile: definirea ferestrei de fundal, cât din fișierul extern se prezintă, dacă redarea se va face după un anumit timp sau dacă redarea trebuie să aștepte un semnal de la un alt eveniment.

În acest capitol veți învăța despre elementele efective ale SMIL, pe parcurs veți crea câteva documente SMIL funcționale și veți vedea cum pot fi folosite elementele SMIL pentru a crea un adevărat spectacol.

Chiar dacă nu vă vine să credeți, sincronizarea multimedia nu este o invenție nouă pe Web. Cei de la HyTime fac acest lucru de ani buni. De fapt, *HyTime* este o prescurtare pentru *Hypermedia Time-based Structuring Language* – limbajul de structurare hipermedia bazat pe timp – și este o arhitectură creată pentru a facilita interacțiunile dintre o mare varietate de documente și instrumente multimedia. Grupul de lucru SMIL s-a bazat pe experiența celor de la HyTime, simplificând și reconsiderând acest limbaj astfel încât SMIL este într-adevăr ușor de folosit și realizează majoritatea lucrurilor pe care le așteptați de la un regizor multimedia.

Sincronizarea

Sincronizarea precisă a unor elemente disparate necesită un anumit standard de timp, care poate fi asigurat fie de la o sursă externă, fie menținut în interiorul sistemului care execută prezentarea sincronizată. Sursele de timp cele mai precise sunt cele externe, conectate într-un anumit mod la standardele de timp păstrate de organisme de standardizare naționale (National Bureau of Standards, SUA, sau multe altele răspândite pe tot globul).

În Statele Unite, ora oficială este furnizată de Observatorul Naval al SUA (USNO) din Washington D.C.; există însă o mulțime de surse secundare. Multe universități și organizații private păstrează propriile cronometre, pe care le sincronizează periodic cu cel al USNO, deci oricine poate găsi ușor un timp standard undeva în vecinătatea sa. Sistemul de navigație Loran C transmite un semnal de timp bazat pe cronometrul USNO în toate regiunile de coastă din lume și în regiunile care cuprind marile fluvii sau lacurile navigabile din interiorul continentelor.

Sateliții GPS (Global Positioning System – sistemul de poziționare globală) transmit semnale de timp ce pot fi detectate de orice receptor GPS care poate să vadă cerul, oriunde pe glob. Precizia semnalelor GPS este păstrată în cadrul a 200 nanosecunde pentru utilizatorii guvernamentali, iar un algoritm de fluctuații aleatoare degradează semnalul la o precizie de 340 de nanosecunde pentru majoritatea utilizatorilor civili. Deoarece fluctuațiile sunt aleatoare, se pot folosi tehnici de aproximare în combinație cu un cronometru local pentru a obține o sincronizare foarte apropiată de cronometrul USNO (Observatorul Naval al SUA).

De asemenea, peste tot în lume există stații radio de înaltă frecvență care transmit semnale de timp, așa încât sunt puține locuri pe glob – poate doar în fundul puțurilor de mină – în care nu se pot recepționa semnale radio de timp.

Receptoarele radio simple și relativ ieftine permit chiar persoanelor și organizațiilor cu mijloace mai modeste accesul la un timp standard, doar cu puțin mai imprecis decât cel al cronometrului USNO sau al altor cronometre similare păstrate de majoritatea națiunilor de pe glob. Aceste surse de timp sunt foarte precise și pot fi utilizate din aproape orice loc de pe glob. Utilizatorii care nu au nevoie de o precizie foarte ridicată pot folosi legăturile telefonice sau Internet ale surselor terestre. Sursele cele mai precise sunt GPS și sistemul de navigație Loran C, dar acest grad de precizie nu este necesar pentru orice sit.

Receptoarele radio utilizează adesea un port de ieșire pentru a transmite fluxul de date unui calculator, de exemplu ASCII pe 7 biți pe o legătură RS-232c, pentru portul serial al calculatorului dumneavoastră. Cum am amintit mai devreme, este posibil să receptați semnale de timp pe Internet utilizând un software ieftin sau chiar gratuit, cum ar fi Cyberkit, Tardis 2000 sau YATS. Întârzierea semnalului într-o rețea limitează precizia la aproximativ 300 de milisecunde, destul de bună pentru multe aplicații, iar prin tehnici statistice se poate îmbunătăți precizia până aproape de (dar nu egală cu) precizia receptoarelor de satelit GPS.

Sugestie de la LeeAnne.com Words to weave by

Sursele on-line, cum este TUCOWS (`http://www.tucows.com/`), sunt surse potrivite pentru software de sincronizare pe Internet, şi există o sumedenie de candidaţi excelenţi, dar unele produse oferă sincronizarea de timp ca o componentă mică într-un sistem mai mare. Un bun exemplu este Cyberkit, care este postcardware (adică autorul cere ca plată pentru utilizarea produsului o carte poştală interesantă) şi care oferă NSLookup, keep-alive, ping, sincronizarea temporală a reţelei şi alte funcţii pentru reţea. Puteţi căuta acest nume direct la TUCOWS; este uşor de găsit.

Rețelele depind de o sincronizare temporală precisă, deci oricine este conectat la o rețea de mare viteză are acces la un cronometru precis, dacă știe cum poate să extragă datele. Aceasta este doar o parte a problemei. Cealaltă parte este păstrarea unor înregistrări precise ale duratelor fișierelor multimedia. Timpul cu precizia cea mai scăzută, numit NPT (Normal Play Time – timp de redare normal) în SMIL, se obține prin măsurarea directă a fișierelor. Cel mai precis este SMPTE (Society of Motion Picture and Television Engineers – asociația tehnicienilor din cinematografie și televiziune), care marchează efectiv sursa din punct de vedere temporal și vă permite accesul la cadre individuale și subcadre. Un cadru este aproximativ a treizecea parte dintr-o secundă, iar un subcadru este măsurat în sutimi de cadru, deci, într-adevăr, puteți specifica extrem de precis începutul și sfârșitul unui fișier. Fișierele codate SMPTE nu sunt foarte folosite pe Web, deci în timp ce NPT este folosit adesea pentru orice conținut, SMPTE necesită un echipament special pentru codare și semnale de timp precise chiar în interiorul conținutului multimedia.

Ca în majoritatea situațiilor din viață, cel mai bine este să începeți planificarea unei producții multimedia complexe cu un creion și o foaie de hârtie. Profesioniștii încep de obicei cu o schemă de scenariu (storyboard), în care sunt precizate scenele și tranzițiile dintre scene într-o manieră neprelucrată, însoțite de note legate de efectele audio necesare. Realizând aceste lucruri pe hârtie – uneori pe foi de hârtie milimetrică pentru precizie de timp, dar la început pe foi detașabile, care sunt comod de manipulat – procesul creativ este facilitat, fără a se pierde prea mult timp cu prelucrarea pe calculator. Un efort de detaliere care apare prea devreme ne blochează procesele de gândire și fixează ideile înainte ca acestea să se dezvolte suficient.

După ce aveți o schemă de scenariu care vă mulțumește, pe dumneavoastră sau pe clienții dumneavoastră, a sosit timpul să începeți completarea detaliilor. Hârtia milimetrică este excelentă pentru vizualizarea unor intervale de timp scurte, și nu vă costă nimic să ștergeți și să modificați o intrare sau o ieșire. Există *numeroase* cărți cu titluri ca *Jump Start Your Brain* de Doug Hall, *Orbiting the Giant Hair Ball* de Gordon MacKenzie, *A Whack on the Side of the Head* de Roger von Oech, *Lateral Thinking* de Edward de Bono și *AHA!* de Jordan Ayan, care vă pot ajuta să deveniți mai creativ, sau cel puțin să abordați creativitatea într-o manieră mai creativă, dar conceptele de bază sunt întâmplarea și fluiditatea. Nu puteți lucra nici foarte fluid și nici prea la întâmplare dacă utilizați ideile altcuiva despre modul de lucru și de organizare a gândurilor.

Utilizarea instrumentelor multimedia pentru crearea documentelor SMIL

Acestea fiind spuse, să presupunem că ați parcurs toate etapele creative și că aveți o idee precisă despre ceea ce doriți să faceți. Utilizarea unor instrumente automate este logică în acest moment, deoarece acum veți adapta ideile și implementările altcuiva la dorințele dumneavoastră. Este o chestiune de măiestrie.

Există destul de multe instrumente multimedia în lume, deși marea lor majoritate sunt patentate. Macromedia, de exemplu, dispune de mai multe seturi, dar toate cer plug-in-uri Macromedia pentru a permite utilizatorului să le vizualizeze. Un standard open-source cum este SMIL presupune că nu sunteți legat de câteva platforme cu porturi ale motorului de redare gata pregătite, ci puteți face aproape tot ce doriți, fiind sigur că aproape orice platformă va putea să vă utilizeze prezentarea.

Microsoft dispune de propriul produs patentat, NetShow, care are câteva caracteristici interesante, dar atunci când îl utilizați sunteți limitat la ceea ce consideră o singură companie și o singură sursă că merită să fie făcut. Deci nu aș conta prea curând pe o portare NetShow, și nici chiar PowerPoint, în Linux, BeOS sau Amiga. Chiar și termenele de producție pentru versiunile Windows ale Mac OS sunt prelungite cu câteva luni sau chiar ani.

SMIL – The Synchronized Multimedia Integration Language – limbajul de integrare multimedia sincronizat

SMIL poate fi implementat în Java, sau în alte limbaje open source, și poate să ruleze oriunde. W3C își actualizează periodic propriul browser Amaya cu ultimele noutăți din seiful lor, iar sursele sunt disponibile gratuit, deci se pot realiza destul de ușor portări pe alte sisteme, chiar individual, cu ajutorul unui compilator sau al unui motor în timpul execuției.

RealSlideshow de la RealNetworks folosește SMIL ca schemă nepatentată, îmbunătățindu-și șansele de funcționare cu viitoarele programe de redare și editare multimedia și cu viitoarele motoare de creare. Produsul existent generează fișiere SMIL și folosește RealPlayer ca motor de redare. Puteți studia oferta RealNetworks la adresa `http://www.real.com/`. Deși încorporează caracteristici RealNetworks patentate în fișierele externe ale prezentărilor create cu RealSlideshow, reușesc să redea destul de bine, dar totuși cu unele limitări, fișierele SMIL 1.0 obișnuite. Figura 23.1 vă arată o prezentare creată manual și rulată de motorul de redare de bază RealPlayer 7.

Sugestie de la LeeAnne.com Words to weave by

Dacă vă gândiți la tehnologiile multimedia patentate, cum sunt Flash, Netshow şi Director, ca la nişte echivalente multimedia ale Microsoft Word sau WordPerfect, care folosesc în mod esenţial formate binare pentru a codifica şi păstra datele, SMIL este ca un editor de texte ASCII ce creează fişiere care nu mai trebuie convertite. În majoritatea cazurilor, pentru formatele multimedia patentate nici măcar nu există programe de conversie.

Prezentarea manipulează destul de bine imaginile, dar nu prea se descurcă la elementele `audio` și `textstream`. Tipurile de fișiere MIDI nu sunt suportate perfect, deci a fost necesar să fie înlocuite cu fișiere WAV. În schimb, fișierele `textstream` nu sunt suportate deloc, deci instrucțiunea a fost inclusă într-un comentariu în acest exemplu. Cu puțin efort în plus, aceste fișiere puteau fi importate în tipuri multimedia native ale RealPlayer, dar astfel am înlocui ipoteza cu concluzia. Un suport complet este evident mai bun decât nici un suport, dar puține companii sunt dispuse să renunțe la avantajele care decurg din patente și să lase pe oricine să scrie fișiere pentru motoarele lor de redare, fără o penalizare în caracteristici și performanță.

FIGURA 23.1
RealPlayer vă arată o prezentare SMIL creată manual după codul SMIL.

Codul principal este exact același cu cel folosit în fișierul exemplu din secțiunea „Corpul documentului" din acest capitol. Singura diferență reală este că numele de fișiere din exemplu au fost înlocuite cu numele unor fișiere reale.

Apple Quicktime 4.1 este un alt produs existent care folosește SMIL. Produsele Apple pot fi găsite la adresa `http://www.apple.com/`.

Explorarea sintaxei SMIL

Secțiunile următoare vă prezintă pe scurt elementele SMIL, deci puteți afla cu ce veți lucra în continuare. Nu sunt chiar foarte multe etichete, și pentru a realiza ceva util aveți nevoie numai de câteva dintre ele, cel puțin pentru început.

Elementul `smil`

Elementul `smil` este rădăcina unui document SMIL și poate primi un atribut unic `id="value"`. Poate să conțină două elemente copil, `head` și `body`. Antetul documentului cuprinde mai ales meta-informații despre mediul de redare, în timp ce corpul documentului realizează efectiv operațiile prezentării. Mai întâi să studiem detaliat antetul documentului.

Antetul documentului

Antetul documentului conține informații despre caracteristicile fizice exacte ale mediului în care va fi redat conținutul documentului SMIL și unele metadate despre document, cum ar fi autorul, data creării și așa mai departe. Nu conține informații de sincronizare temporală de nici un fel.

Elementul `head` Elementul `head` poate să primească un atribut `id`, poate să conțină orice număr de descendenți `meta` și poate să aibă fie un singur descendent `layout`, fie un singur descendent `switch`, care poate să conțină o listă exclusivă de descendenți `layout`.

Elementul `layout` Elementul `layout` poate să primească un atribut `id` și un atribut `type`, care definește limbajul de machetare folosit. Deși SMIL este un astfel de limbaj, nu este limitat la el însuși, ci poate să apeleze la facilitățile unui alt limbaj pe care îl înțelege sau pentru care dispune de o aplicație ajutătoare. Dacă tipul este `text/smil-basic-layout`, atunci elementul `layout` poate să conțină elementele `region` și `root-layout`. În caz contrar, elementul `layout` poate să conțină orice text. De exemplu, în loc să folosiți facilitățile de machetare ale SMIL, ați putea folosi CSS2 împreună cu SMIL, pentru a permite agentului utilizator să își aleagă mecanismul pe care îl va folosi pentru redarea conținutului, astfel:

```
<head>
  <switch title="[Set the banner display window position]">
    <layout type="text/css">
      [region="banner"] { top: 10px; left: 10px }
    </layout>
    <layout>
      <region id="banner" title="[Banner]" top="10" left="10" />
    </layout>
  </switch>
</head>
```

Observați că SMIL dublează multe din facilitățile de machetare ale CSS folosind o sintaxă de bază similară. Pentru simplitate, nu sunt prezentate toate elementele de machetare pe care ați putea să le specificați, ci doar cele esențiale, dimensiunile imaginilor definind dimensiunea ferestrei care va apărea pe ecran. Acesta poate fi un mod în care să redați o prezentare, dar mai sunt și alte posibilități. Veți învăța mai multe despre modalitățile în care puteți controla plasarea exactă și dimensiunile regiunilor de pe ecran în secțiunea următoare, „Limbajul de machetare de bază al SMIL".

Limbajul de machetare de bază al SMIL

Limbajul de machetare de bază al SMIL este aproape un duplicat al celor mai des folosite proprietăți ale CSS2. Principala adăugare este atributul `fit`, care în CSS2 poate fi numai simulat, dar este mult mai ușor de manipulat ca simplu atribut. Limbajul este alcătuit din atribute asociate celor două elemente de machetare ale SMIL din `head`, elementele `region` și `root-layout`. În timp ce elementele `root-layout` definesc fereastra globală de redare a prezentării, elementele `region` împart opțional această fereastră în atâtea segmente mai mici câte sunt necesare. Dacă doriți, puteți utiliza și întreaga fereastră.

Dacă vă gândiți la un document SMIL ca la o serie de evenimente similare cu cele pe care le-ați vedea într-un teatru, elementele `root-layout` și `region` ar caracteriza avanscena și scena și ar defini anumite locuri pe scenă, astfel încât să puteți plasa actorii (chiar elementele multimedia) la locul potrivit. Romeo este în grădină, iar Julieta este în balconul de deasupra, și le puteți coordona acțiunile ca un regizor care planifică acțiunile și replicile actorilor. Observați că regiunile definite pentru anumite acțiuni pot fi regiuni vizibile pe ecran sau regiuni acustice, definite prin plasarea acustică realizată cu tehnologii de sunet stereo sau chiar surround.

Elementul `region`

Elementul `region` nu are conținut, dar controlează poziția, dimensiunea și scalarea elementelor obiecte multimedia. Elementul primește un atribut `id` obligatoriu, care vă permite să selectați elementele obiecte multimedia ale regiunii care vor fi afișate cu un atribut `region` corespunzător. Elementele obiecte multimedia apar numai în corpul documentului SMIL și sunt `ref`, `animation`, `audio`, `img`, `video`, `text` și `textstream`.

▶ Pentru a afla mai multe despre informaţiile conţinute în elementul body al documentului, **vezi** „Corpul documentului", în această pagină.

Elementul region poate să primească următoarele atribute:

- **background-color** – Foarte asemănător cu background-color din CSS, cu excepţia faptului că numele de culori pot să fie suportate sau nu.
- **fit** – Descrie modul în care conţinutul multimedia se potriveşte în aria definită de regiune. Valorile posibile sunt fill, care dimensionează conţinutul astfel încât să se potrivească indiferent de aspect; meet, care dimensionează conţinutul astfel încât să se potrivească în regiune, dar menţine aspectul original, deci este posibil să rămână goluri la dreapta sau dedesubt; scroll, care apelează la un anumit mecanism de defilare în cazul în care conţinutul multimedia este mai mare decât fereastra de afişare a regiunii; slice, care nu face acest lucru, şi hidden, care afişează conţinutul multimedia ca şi slice, dar umple spaţiile libere cu culoarea de fundal. Valoarea prestabilită este hidden.
- **height** – Este identic cu proprietatea CSS corespunzătoare.
- **left** – Este identic cu proprietatea CSS corespunzătoare.
- **skip-content** – Acest atribut boolean permite procesoarelor SMIL 1.0 să ignore conţinutul elementului region dacă o specificare ulterioară îl modifică de la un element vid la unul cu conţinut.
- **title** – O descriere a regiunii lizibilă pentru utilizator. Deşi nu este obligatorie, este foarte recomandată.
- **top** – Este identic cu proprietatea CSS corespunzătoare.
- **width** – Este identic cu proprietatea CSS corespunzătoare.
- **z-index** – Este identic cu proprietatea CSS corespunzătoare.

Elementul root-layout

Acest element nu are conţinut, dar controlează dimensiunile elementului root, care defineşte fereastra în care documentele referite de SMIL sunt prezentate. Primeşte atributele id şi title, aşa cum am văzut mai înainte, şi, de asemenea, atributele height, width, top şi left, care sunt suficiente pentru a defini o fereastră dreptunghiulară. Atributul boolean skip-content permite procesoarelor SMIL 1.0 să ignore conţinutul elementului root-layout dacă o specificare ulterioară îl modifică de la un element vid la unul cu conţinut.

Elementul meta

Elementul meta nu are conţinut, dar descrie o listă deschisă de proprietăţi care se aplică documentului ca întreg. Primeşte atributele obligatorii name şi content, care descriu numele, respectiv valoarea proprietăţii. Poate primi opţional şi atributele id şi skip-content care au fost descrise mai sus.

Câteva dintre numele de proprietăţi predefinite sunt base, care este URI-ul de bază pentru URI-uri relative; title, titlul prezentării, şi pics-label (sau PICS-Label), care se referă la standardul PICS descris în Capitolul 27, „ Concluzii – viitorul în Web".

Corpul documentului

Corpul documentului conţine toate informaţiile dependente de timp şi toate informaţiile de legătură asociate documentului. Elementul head pregăteşte scena, ca să zicem aşa,

asigurând cadrul de bază în care se va desfăşura spectacolul multimedia. În schimb, intrările şi ieşirile fiecărui actor care apare pe această scenă sunt controlate de elementul body. Aşa cum o piesă de teatru are un regizor, un spectacol SMIL are un element body care spune fiecărui actor extern multimedia al spectacolului când şi unde să apară, cât timp să apară şi ce rol să joace.

De exemplu, următorul fragment SMIL va iniţia o prezentare simplă, cu titluri nesincronizate şi cu un fond muzical:

```
<smil>
  <head>
    <layout>
      <root-layout id="SlidesShow" title="[Slide Show Window]"
                   width="400" height="400" />
      <region id="slides" title="[Slide Show]" top="10" left="10" />
      <region id="sound" title="[Audio Track]" />
    </layout>
  </head>
  <body>
    <par>
      <audio src="musicfile" region="sound" />
      <textstream src="captionfile" />
      <seq>
        <img src="slide1file" region="slides" dur="10s" />
        <img src="slide2file" region="slides" dur="10s" />
        <img src="slide3file" region="slides" dur="10s" />
        <img src="slide4file" region="slides" dur="10s" />
        <img src="slide5file" region="slides" dur="10s" />
      </seq>
    </par>
  </body>
</smil>
```

Puteţi înlocui numele de fişiere generice din această schemă cu fişierele pe care le aveţi, şi ar trebui să funcţioneze destul de bine. Pentru a vă asigura în marea diversitate a motoarelor de redare, fişierele WAV sunt destul de generale, deşi în paginile Web se folosesc de obicei fişiere MIDI. Dimensiunea ferestrei, definită în elementul root-layout, şi plasamentul imaginilor efective, definit în elementul region, pot fi reglate pentru a se potrivi cu fişierele dumneavoastră. Veţi învăţa mai multe despre elementele folosite pentru coordonarea prezentării în secţiunea următoare. Aţi putut vedea prezentarea în Figura 23.1.

Elementul body

Elementul body primeşte un atribut id şi poate să conţină următorii descendenţi opţionali:

- **a** – Un element ancoră echivalent cu un element HTML linktext.
- **animation** – Identifică fişierul referit ca fişier de animaţie.
- **audio** – Identifică fişierul referit ca fişier audio. Observaţi că o suprafaţă de redare audio nu poate fi definită prin macheta de bază SMIL, deoarece pentru definirea suprafeţelor acustice este nevoie de CSS2 sau de o variantă mai bună.
- **img** – Identifică fişierul referit ca fişier imagine.
- **par** – Îi spune motorului SMIL să redea descendenţii pe care îi conţine în paralel, adică simultan.

- **ref** – Îi spune motorului SMIL că fişierul referit este un obiect multimedia general. Acest element este folosit atunci când, din diferite motive, celelalte tipuri multimedia nu se potrivesc.
- **seq** – Îi spune motorului SMIL să redea descendenţii pe care îi conţine secvenţial, adică unul după celălalt.
- **switch** – Îi spune motorului SMIL că trebuie să examineze conţinutul din interiorul său.
- **text** – Identifică fişierul referit ca fişier text, adică un fişier text care poate fi afişat într-o singură fereastră.
- **textstream** – Identifică fişierul referit ca fişier `textstream`, adică un fişier text care trebuie să defileze printr-o fereastră.
- **video** – Identifică fişierul referit ca fişier video.

Elemente de sincronizare

Într-o formulare simplă, elementele de sincronizare sunt cele două elemente de sincronizare de bază, `par` şi `seq`, şi elementele obiecte multimedia, care sunt elementele `ref`, `animation`, `audio`, `img`, `video`, `text` şi `textstream`.

Elementul par

Descendenţii unui element `par` formează un grup de evenimente multimedia care nu sunt foarte strâns legate şi care se pot suprapune, sau pot să se execute simultan. Manipularea exactă a momentelor de început şi de sfârşit şi a duratei poate duce la variaţii ale efectelor generate care sunt limitate numai de imaginaţia dumneavoastră.

Elementul `par` poate primi următoarele atribute:

- **abstract** – O descriere scurtă a conţinutului din interiorul elementului.
- **author** – Numele autorului conţinutului din interiorul elementului.
- **begin** – Acest atribut specifică momentul de început al unui element.
- **event-value** – Elementul începe în momentul în care apare un anumit eveniment.
- **copyright** – Nota de copyright a conţinutului din interiorul elementului.
- **dur** – Acest atribut specifică durata explicită a unui element.
- **end** – Acest atribut specifică sfârşitul explicit al unui element.
- **endsync** – Acest atribut poate să aibă următoarele valori: `first`, `id-ref`, `last`. Vă permite să sincronizaţi sfârşitul redării unui element multimedia cu un alt eveniment.
- **id** – Acest atribut identifică un element cu un identificator unic şi nimic mai mult.
- **region** – Acest atribut specifică o suprafaţă de redare abstractă.
- **repeat** – Valoarea acestui atribut poate fi un număr întreg sau şirul „`indefinite`". Valoarea prestabilită este `1`.
- **system-bitrate** – Evaluează rata de transfer declarată de un agent utilizator faţă de rata de transfer cerută de un anumit element. Aceasta vă permite să încărcaţi fişiere mai voluminoase în cazul unei conexiuni de mare viteză şi fişiere mai firave în cazul unor linii telefonice cu viteză redusă.
- **system-captions** – În funcţie de setările `on` şi `off`, se alege afişarea legendelor elementelor aşa cum doreşte utilizatorul. Deoarece există două atribute care se referă la legende, acestea pot fi folosite în combinaţii logice destul de sofisticate.

- **system-language** – În funcţie de limbile disponibile într-o listă a agentului utilizator, este aleasă limba cea mai potrivită pentru afişare sau redare.
- **system-overdub-or-caption** – În funcţie de setările `overdub` şi `caption`, se alege redarea unei secvenţe audio descriptive sau afişarea legendelor elementelor, aşa cum doreşte utilizatorul. Deoarece există două atribute care se referă la legende, acestea pot fi folosite în combinaţii logice destul de sofisticate.
- **system-required** – Specifică numele unei extensii la funcţionalitatea de bază a SMIL. Dacă agentul utilizator suportă această extensie, elementul care o cere este ales. Este posibil ca aici să apară de fapt o zonă de nume XML.
- **system-screen-size** – În funcţie de dimensiunea ecranului raportată de agentul utilizator, se alege o dimensiune potrivită pentru regiune şi pentru fişierele multimedia.
- **system-screen-depth** – În funcţie de adâncimea de culoare raportată a agentului utilizator, se aleg fişierele multimedia cele mai potrivite pentru afişare.
- **title** – Un titlu pentru element, care poate să fie redat sau nu, în funcţie de agentul utilizator şi de cerinţele particulare ale utilizatorului.

Elementul `par` poate să conţină următorii descendenţi, deja definiţi:

- `a`
- `animation`
- `audio`
- `img`
- `par`
- `ref`
- `seq`
- `switch`
- `text`
- `textstream`
- `video`

Elementul seq

Descendenţii unui element `seq` formează o secvenţă temporală în care un eveniment urmează după altul într-o ordine strictă. Deoarece evenimentele transmise pe o reţea pot să ajungă în dezordine la destinaţie, o dependenţă exclusivă de momentele de început şi de sfârşit poate să ducă la suprapuneri deranjante. Când este obligatoriu ca elementele unei prezentări să nu se suprapună, utilizarea unui element `seq` pentru a le controla face ca SMIL să garanteze că acestea nu se vor suprapune, chiar dacă un element este întârziat în transmisie.

Elementul `seq` poate primi următoarele atribute:

- **abstract** – O descriere scurtă a conţinutului din interiorul elementului.
- **author** – Numele autorului conţinutului din interiorul elementului.
- **begin** – Acest atribut specifică momentul de început al unui element.
- **copyright** – Nota de copyright a conţinutului din interiorul elementului.
- **dur** – Acest atribut specifică durata explicită a unui element.
- **end** – Acest atribut specifică sfârşitul explicit al unui element.
- **id** – Acest atribut identifică un element cu un identificator unic.
- **region** – Acest atribut specifică o suprafaţă de redare abstractă.

- **repeat** – Valoarea acestui atribut poate fi un număr întreg sau șirul „indefinite". Valoarea prestabilită este 1.
- **system-bitrate** – Evaluează rata de transfer declarată de agentul utilizator față de rata de transfer cerută de un anumit element. Aceasta vă permite să încărcați fișiere mai voluminoase în cazul unei conexiuni de mare viteză și fișiere mai firave în cazul unor linii telefonice cu viteză redusă.
- **system-captions** – În funcție de setările on și off, se alege afișarea legendelor elementelor așa cum dorește utilizatorul. Deoarece există două atribute care se referă la legende, acestea pot fi folosite în combinații logice destul de sofisticate.
- **system-language** – În funcție de limbile disponibile într-o listă a agentului utilizator, este aleasă limba cea mai potrivită pentru afișare sau redare.
- **system-overdub-or-caption** – În funcție de setările overdub și caption, se alege redarea unei secvențe audio descriptive sau afișarea legendelor elementelor, așa cum dorește utilizatorul. Deoarece există două atribute care se referă la legende, acestea pot fi folosite în combinații logice destul de sofisticate.
- **system-required** – Specifică numele unei extensii la funcționalitatea de bază a SMIL. Dacă agentul utilizator suportă această extensie, elementul care o cere este ales. Este posibil ca aici să apară de fapt o zonă de nume XML.
- **system-screen-size** – În funcție de dimensiunea ecranului raportată de agentul utilizator, se alege o dimensiune potrivită pentru regiune și pentru fișierele multimedia.
- **system-screen-depth** – În funcție de adâncimea de culoare raportată a agentului utilizator, se aleg fișierele multimedia cele mai potrivite pentru afișare.
- **title** – Un titlu pentru element, care poate să fie redat sau nu, în funcție de agentul utilizator și de cerințele particulare ale utilizatorului.

Elementul seq poate să conțină următorii descendenți, deja definiți:

- a
- animation
- audio
- img
- par
- ref
- seq
- switch
- text
- textstream
- video

Elementele obiecte multimedia

Elementele obiecte multimedia, care cuprind elementele ref, animation, audio, img, video, text și textstream, permit includerea prin referință a obiectelor media într-o prezentare SMIL și primesc următoarele atribute:

- **abstract** – O descriere scurtă a conținutului din interiorul elementului.
- **alt** – O descriere alternativă pentru agenții care nu pot reda conținutul multimedia sau un ajutor pentru persoanele care nu îl pot utiliza într-o anumită situație.
- **author** – Numele autorului conținutului din interiorul elementului.

- **begin** – Acest atribut specifică momentul de început al unui element.
- **event-value** – Elementul începe în momentul în care apare un anumit eveniment.
- **clip-begin** – Specifică momentul de început ca offset față de începutul unui fișier multimedia folosind SMPTE sau NPT.
- **clip-end** – Specifică momentul de sfârșit ca offset față de începutul unui fișier multimedia folosind SMPTE sau NPT.
- **copyright** – Nota de copyright a conținutului din interiorul elementului.
- **dur** – Acest atribut specifică durata explicită a unui element.
- **end** – Acest atribut specifică sfârșitul explicit al unui element.
- **fill** – Poate fi remove sau freeze. Acest atribut controlează comportarea unui element par sau a unui element seq terminal care se încheie înaintea unui element care îl conține sau a unui element seq care nu este terminal, dar care se încheie înaintea începutului următorului element. Valoarea remove spune procesorului să închidă fișierul, iar valoarea freeze îi spune să facă un stop-cadru până la încheierea elementului părinte sau până la începutul următorului element.
- **longdesc** – Un URI care indică o descriere mai lungă ce completează descrierea alternativă din atributul alt.
- **id** – Acest atribut identifică un element cu un identificator unic.
- **region** – Acest atribut specifică o suprafață de redare abstractă.
- **src** – URI-ul care identifică sursa fișierului multimedia.
- **system-bitrate** – Evaluează rata de transfer declarată de un agent utilizator față de rata de transfer cerută de un anumit element. Aceasta vă permite să încărcați fișiere mai voluminoase în cazul unei conexiuni de mare viteză și fișiere mai firave în cazul unor linii telefonice cu viteză redusă.
- **system-captions** – În funcție de setările on și off, se alege afișarea legendelor elementelor așa cum dorește utilizatorul. Deoarece există două atribute care se referă la legende, acestea pot fi folosite în combinații logice destul de sofisticate.
- **system-language** – În funcție de limbile disponibile într-o listă a agentului utilizator, este aleasă limba cea mai potrivită pentru afișare sau redare.
- **system-overdub-or-caption** – În funcție de setările overdub și caption, se alege redarea unei secvențe audio descriptive sau afișarea legendelor elementelor așa cum dorește utilizatorul. Deoarece există două atribute care se referă la legende, acestea pot fi folosite în combinații logice destul de sofisticate.
- **system-required** – Specifică numele unei extensii la funcționalitatea de bază a SMIL. Dacă agentul utilizator suportă această extensie, elementul care o cere este ales. Este posibil ca aici să apară de fapt o zonă de nume XML.
- **system-screen-size** – În funcție de dimensiunea ecranului raportată de agentul utilizator, se alege o dimensiune potrivită pentru regiune și pentru fișierele multimedia.
- **system-screen-depth** – În funcție de adâncimea de culoare raportată a agentului utilizator, se aleg fișierele multimedia cele mai potrivite pentru afișare.
- **title** – Un titlu pentru element, care poate să fie redat sau nu, în funcție de agentul utilizator și de cerințele particulare ale utilizatorului.
- **type** – Tipul MIME al obiectului multimedia referit prin atributul src.

Toate obiectele multimedia pot să conțină un element copil ancoră, astfel încât să puteți construi hiperlegături folosind ancora, adică eticheta <a>.

Modelul temporal al SMIL

Dacă vă gândiţi la toate lucrurile care pot să meargă prost în cazul transmisiilor multimedia prin Internet, sigur vă daţi seama că timpul este un lucru destul de alunecos. Posibilităţile de întârziere sunt nesfârşite, aşa cum poate să vă spună oricine a încercat vreodată să intre pe Web cu un modem telefonic, deci ce se întâmplă atunci când lucrurile încetinesc la o viteză de melc? Cel mai scurt răspuns este „Depinde". Problemele de temporizare pot fi extrem de complexe şi variază în funcţie de modul exact în care a fost specificat timpul. Recomandarea SMIL 1.0 dedică mai multe pagini, destul de neclare, examinării tuturor posibilităţilor şi ajunge cam la aceeaşi concluzie. Adevărul este că uneori lucrurile *vor merge* prost şi nu prea aveţi ce să faceţi decât să treceţi pe o conexiune Internet mai rapidă.

Ceea ce, apropo, vă recomand cu căldură. În majoritatea cazurilor nu este chiar atât de costisitor să treceţi la o conexiune DSL sau la un serviciu prin modem pe cablu, mai ales dacă vă gândiţi cât timp veţi economisi. Orice persoană care a cumpărat această carte probabil că lucrează destul de mult pe Web, aşa încât investiţia va fi recuperată înzecit. Deşi linia dumneavoastră de la serviciu poate să fie una T1 sau T3, ciudăţeniile vieţii de informatician vă vor obliga din când în când să consumaţi timp pentru a rezolva mici urgenţe în miez de noapte. Ar fi mult mai frumos să nu trebuiască să vă îmbrăcaţi şi să vă duceţi până la serviciu pentru a rezolva o problemă căreia îi puteţi da de capăt în zece sau douăzeci de minute on-line.

Valorile modelului temporal

Există două tipuri de valori de timp care se folosesc în SMIL. NPT (Normal Playing Time – timp de redare normal) vă permite să măsuraţi timpul de redare de la începutul unui clip şi să notaţi momentele exacte în care doriţi să înceapă şi să se termine subclipul. SMPTE a fost inventat de tehnicienii din televiziune şi cinematografie şi este puţin mai complicat decât NPT, dar este extrem de precis. Staţiile de lucru profesionale pentru editarea video măsoară timpul conform SMPTE, deoarece *timpul* exact este mai puţin important decât *cadrul* exact al unui film sau al unui videoclip.

În codul dumneavoastră, NPT va avea următorul format:

```
HH:MM:SS.hh
```

unde HH este un număr de ore, MM este între 0 şi 59 de minute, SS este între 0 şi 59 de secunde, iar hh sunt sutimi de secundă. Dacă cifrele cu rang mai mare sunt ø, ele pot fi omise. Dacă este specificată unitatea de măsură, un interval numeric va fi atât de mare cât este necesar pentru a exprima perioada respectivă. `npt=128.23s` este echivalent cu `npt=02:08.234`.

SMPTE este ceva mai complex. Mai întâi, există trei tipuri diferite de SMPTE pentru a se potrivi cu cerinţele diferiţilor utilizatori. `smpte` foloseşte 30 de cadre pe secundă, în timp ce `smpte-30-drop` foloseşte 23,97 cadre pe secundă printr-un aranjament complicat în care cadrele sunt sărite la câteva minute. `smpte-25` foloseşte 25 de cadre pe secundă. Toate arată la fel, dar diferă prin numărul de cadre care poate fi completat în câmpul cadrelor.

Sugestie de la LeeAnne.com Words to weave by

SMPTE, Society of Motion Picture and Television Engineers – asociaţia tehnicienilor din cinematografie şi televiziune, publică o mulţime de standarde legate de fiecare fază de producţie din televiziune şi cinematografie. Acestea pot fi găsite la `http://www.smpte.org`.

În codul dumneavoastră, SMPTE arată astfel:

```
HH:MM:SS:ff.hh
```

unde HH este un număr de ore, MM este între 0 şi 59 de minute, SS este între 0 şi 59 de secunde, ff este numărul de cadre între 0 şi numărul permis de un anumit format, iar hh sunt sutimi de cadru.

Fiecare dintre aceste standarde are o utilizare particulară şi ele provin din diferitele formate cinematografice şi de televiziune. De fapt, au lăsat pe dinafară `smpte-24`, care foloseşte 24 de cadre pe secundă. Problema este că în clipa de faţă se utilizează mai multe sisteme de televiziune şi, de asemenea, mai multe sisteme cinematografice, astfel încât este mai comod să existe formate de conversie care să fie legate de sursă printr-un coeficient întreg.

Determinarea valorilor de timp pentru elementele SMIL 1.0

Deoarece în SMIL totul depinde atât de mult de temporizare, există mai multe reguli care vă spun exact care sunt valorile de timp implicite în toate situaţiile în care momentele efective de început şi de sfârşit pot fi incerte. Desigur, dacă specificaţi momente exacte pentru fiecare element, problema momentelor implicite dispare, dar această abordare poate să nu se potrivească sau să nu fie practică pentru o prezentare de mari dimensiuni. Uneori, tranziţiile trebuie să fie controlate de utilizator; de exemplu, un cititor nu va trece la următoarea imagine sau la următorul clip până când nu a ajuns într-un anumit punct din textul prezentării. Uneori utilizatorul va dori să încetinească prezentarea pentru a fi mai uşor de înţeles. Iar în unele cazuri, latenţa pe reţea poate întârzia o perioadă nedeterminată începutul unui anumit element.

În toate situaţiile acestea aveţi nevoie de un set de reguli pentru a determina flexibil ce se redă la un anumit moment al prezentării, ce urmează după un element şi după cât timp. Secţiunile care urmează vă prezintă o listă cu aceste reguli şi câteva exemple pentru clarificarea problemelor.

Determinarea începutului implicit al unui element

Dacă nu stabiliţi explicit momentul în care începe un element, acesta va începe la un moment dedus din alte evenimente după următoarele reguli:

- Începutul implicit al primului copil al elementului `body` este momentul în care începe redarea documentului, adică elementele sunt redate imediat şi la cerere, în afară de cazul în care modificaţi explicit acest comportament.
- Începutul implicit al oricărui copil al unui element `par` este acelaşi cu începutul efectiv al elementului `par`. Cu alte cuvinte, toate elementele dintr-o secţiune `par` încep în acelaşi moment, cel puţin teoretic, adică exact aşa cum vă aşteptaţi.
- Începutul implicit al primului copil al unui element `seq` este acelaşi cu începutul efectiv al elementului `seq`, adică, mai simplu spus, primul element dintr-o secvenţă începe imediat, în afară de cazul în care specificaţi altceva.

- Începutul implicit al oricărui alt copil al unui element seq este același cu sfârșitul dorit al elementului copil anterior al elementului seq. Cu alte cuvinte, elementele se succed într-o secvență strictă și fără suprapuneri.

Deci, în codul următor, începutul implicit al ambelor elemente este același, conform regulii 2, deoarece sunt incluse într-un element par:

```
<par>
  <audio src="audiofile1" />
  <audio src="audiofile2" />
</par>
```

În al doilea exemplu, începutul implicit al celui de-al doilea fișier este imediat după sfârșitul primului fișier, conform regulii 3, orice durată ar avea acesta, deoarece se află în interiorul unui element seq:

```
<seq>
  <audio src="audiofile1" />
  <audio src="audiofile2" />
</seq>
```

Dacă ați modificat comportamentul primului element audio din secvență printr-un atribut de sfârșit explicit, începutul implicit al celui de-al doilea element audio este acum imediat după sfârșitul explicit al primului element, conform regulii 3:

```
<seq>
  <audio src="audiofile1" end="12s" />
  <audio src="audiofile2" />
</seq>
```

Nu este chiar atât de complicat. Comportamentul elementelor este destul de intuitiv după ce înțelegeți modul în care elementele multimedia interacționează cu comportamentul dorit de utilizatorii de browsere și cu inconstanța rețelei.

Determinarea sfârșitului implicit al unui element

Dacă nu specificați momentul în care se termină un element, acesta se va termina la un moment care depinde de interacțiunile și relațiile dintre atributele care descriu comportarea elementului însuși și a fișierelor la care se referă elementul.

Prima descriere din lista următoare care se potrivește cu elementul este cea care se aplică. Nu puteți alege la întâmplare, ci trebuie să treceți pas cu pas prin listă:

1. Un element care are un atribut repeat cu valoarea indefinite are sfârșitul implicit imediat după începutul său. Cu alte cuvinte, indefinit înseamnă chiar indefinit. Elementul se poate termina în orice moment, în funcție de un eveniment extern oarecare, dar nu poate să se încheie de la sine.
2. Un element care are un atribut repeat cu orice altă valoare în afară de indefinite are sfârșitul implicit egal cu sfârșitul implicit al unui element seq care ar cuprinde același număr de cópii ale elementului ca și copiii. Deci repetarea unui element este același lucru cu inserarea unei secvențe cu un număr de elemente duplicate egal cu numărul de repetări.
3. Un element multimedia care se referă la un obiect continuu, cum ar fi un fișier audio sau video, are un sfârșit implicit egal cu suma dintre începutul elementului și durata sa. Deci dacă alegeți o sursă cu o durată implicită, cum ar fi un fișier .wav, elementul va dura până la sfârșitul sursei.

4. Un element multimedia care se referă la un fișier fără durată intrinsecă, cum ar fi un fișier text sau o imagine, are sfârșitul implicit imediat după începutul său efectiv. Textele și imaginile sunt afișate instantaneu, sau cel puțin acesta este modul în care sunt concepute.
5. Un element seq are sfârșitul implicit egal cu sfârșitul dorit al ultimului său copil. Deci, dacă stabiliți un sfârșit explicit pentru ultimul element copil, acesta este momentul în care se încheie secvența, chiar dacă nici un fișier nu a fost redat până în capăt.
6. Un element par are un sfârșit implicit care depinde de valoarea unui atribut inclus endsync. Dacă valoarea atributului endsync este last, valoare care este prestabilită și se aplică dacă atributul lipsește, durata implicită a elementului par este egală cu cea mai mare dintre duratele dorite ale elementelor sale copil, deci elementul va continua să ruleze până ce se încheie și ultimul său element copil. În schimb, dacă valoarea atributului endsync este first, atunci durata implicită a elementului par este egală cu cea mai mică dintre duratele dorite ale elementelor sale copil, deci elementul va continua să ruleze până ce se încheie primul său element copil. Iar în cazul în care valoarea atributului endsync este id-ref, atunci durata implicită a elementului par este egală cu durata dorită a elementului copil referit prin id-ref, deci puteți opri elementul atunci când un anumit element copil se încheie.

Deci, în următorul exemplu, durata întregului element par este egală cu durata fișierului audio referit prin ID-ul fileid, conform regulii 6. Chiar dacă durata intrinsecă a fișierului video este mai mare decât a fișierului audio, ambele fișiere se încheie în același moment, deoarece elementul par care le conține se încheie. Dacă durata fișierului video este mai mică decât durata fișierului audio, atunci elementul par continuă să ruleze până la sfârșitul fișierului audio, chiar dacă fișierul video s-a încheiat.

```
<par endsync="id(fileid)">
  <audio id="fileid" src="audiofile" />
  <video src="videofile" />
</par>
```

Determinarea sfârșitului dorit al unui element

Dacă stabiliți fie o durată explicită, fie un sfârșit explicit, înseamnă că definiți comportamentul elementului astfel încât să aibă un sfârșit dorit în loc de un sfârșit implicit. Ca în secțiunea precedentă, se aplică prima situație care se potrivește din lista următoare; adică nu puteți selecta la întâmplare din listă.

1. Dacă un element multimedia are *atât* o durată explicită *cât și* un sfârșit explicit, atunci sfârșitul dorit este valoarea minimă *dintre* suma începutului dorit și a duratei explicite *și* sfârșitul explicit. Dacă una dintre valori este mai mare, este păstrată valoarea mai mică. Cu alte cuvinte, agentul utilizator va determina ceea ce ați dorit într-adevăr să faceți, fie vă place fie nu, deci însumați cu grijă valorile respective.
2. Dacă elementul *are* o durată explicită dar *nu are* un sfârșit explicit, sfârșitul dorit este suma dintre începutul dorit și durata explicită, ceea ce este logic, dacă vă gândiți puțin. Dacă îi cereți să redea ceva pentru o anumită perioadă, asta va face.
3. Dacă elementul *are* un sfârșit explicit dar *nu are* o durată explicită, sfârșitul dorit este exact sfârșitul explicit. Cu alte cuvinte, dacă nu îi dați alte instrucțiuni, afișarea unui element multimedia se încheie atunci când spuneți dumneavoastră.

4. În celelalte cazuri, sfârșitul dorit este egal cu sfârșitul implicit, ceea ce înseamnă că agentul de redare presupune că știți ce faceți și doriți într-adevăr o temporizare controlată prin comportamentul prestabilit.

Determinarea începutului dorit al unui element

Presupunând că doriți să controlați începutul unui eveniment multimedia din prezentarea dumneavoastră, trebuie să inserați un atribut de timp de un anumit tip. În caz contrar, agentul utilizator se va comporta conform regulilor prestabilite. Începutul dorit al unui element este egal cu începutul explicit, dacă acesta există; în caz contrar, începutul dorit este egal cu începutul implicit, ceea ce presupune că știți ce faceți și sunteți pregătit să acceptați comportamentele prestabilite. Specificarea unui început explicit mai devreme de începutul implicit este o eroare, deci fiți atent la temporizare.

Determinarea începutului efectiv al unui element

Începutul efectiv al unui element este egal cu începutul dorit al elementului, în afară de cazul în care sfârșitul efectiv al elementului părinte este mai devreme decât începutul elementului copil, caz în care elementul nici măcar nu este redat. Deci elementul `video` din următorul fragment SMIL nu va fi afișat niciodată, deoarece elementul `par` care îl conține se încheie înainte de începutul segmentului `video`:

```
<par endsync="id(fileid)">
  <audio id="fileid" end="10s" src="audiofile" />
  <video begin="15s" src="videofile" />
</par>
```

Este un fragment perfect logic, dar probabil că în acest caz eroarea a fost făcută în timpul creării documentului SMIL. Un scenariu mai rațional ar arăta cam așa:

```
<par dur="indefinite">
  <audio src="audiofile" />
  <video begin="15s" src="videofile" />
</par>
```

În acest caz, execuția elementului `par` poate fi întreruptă de un eveniment exterior, să zicem intervenția utilizatorului, iar în acest caz atât fișierul audio cât și fișierul video se vor opri dacă deja au început, dar în caz contrar fișierul audio va fi redat o perioadă, după care va începe și redarea fișierului video.

Determinarea sfârșitului efectiv al unui element

Sfârșitul efectiv al unui element este rezultatul mai multor lucruri, majoritatea acestora fiind în afara controlului dumneavoastră. Întârzierile pe rețea sau erorile pot modifica temporizările bufferelor pentru începutul redării, atât viteza conexiunii la rețea a utilizatorului final cât și capacitatea serverului de fișiere de a furniza fișierele cerute pot afecta afișarea, și chiar agentul utilizator în sine poate produce întârzieri. Mulți agenți utilizatori încorporează caracteristici de pauză sau de încetinire automată a redării pentru a răspunde unor nevoi speciale. Metoda prin care elementele sunt sincronizate, dacă sunt măcar sincronizate, depinde chiar de agentul utilizator. Nu puteți să contați pe rezultate perfecte într-o lume imperfectă.

Regulile care urmează se referă la modul în care apar pe ecran elementele în momentele lor finale dacă alte elemente sunt întârziate din anumite motive. În fond, alegerea este între stop cadru până la sfârșitul efectiv sau o dispariție discretă.

- Sfârșitul efectiv al ultimului element copil al elementului `body` depinde de agentul utilizator. Sfârșitul efectiv al ultimului element copil, și deci al întregii prezentări, este cel puțin la fel de târziu ca și sfârșitul dorit, dar depinde de implementare dacă întârzie mai mult. Cu alte cuvinte, nu puteți conta pe un anumit moment de sfârșit decât dacă știți cum se comportă un anumit agent utilizator.
- Dacă oricare dintre copiii unui element `par` are un atribut `fill` cu valoarea `freeze`, sfârșitul efectiv al elementului copil este egal cu sfârșitul efectiv al elementului părinte, chiar dacă durata intrinsecă a elementului este mai mică decât a părintelui, iar ultima stare a elementului respectiv este păstrată pe ecran până la sfârșitul său efectiv. Poate că acest comportament al afișării vă convine sau nu, dar oricum aceasta este regula. Este o alegere destul de logică dacă, să zicem, o secvență audio descrie o secvență video și este evident mai normal să aveți o imagine oprită pe ecran decât să aveți o secvență audio care vorbește vrute și nevrute despre ceea ce ar trebui să vedeți când dumneavoastră vedeți de fapt un ecran alb.
- Dacă oricare dintre copiii unui element `par` are un atribut `fill` cu valoarea `remove`, sfârșitul efectiv al elementului copil este cel mai mic dintre sfârșitul efectiv al elementului părinte și sfârșitul dorit al elementului copil. Cu alte cuvinte, dacă sfârșitul elementului părinte este întârziat din anumite motive, elementul copil se va comporta frumos și va dispărea oricum.
- Dacă un copil al unui element `par` nu are un atribut `fill`, sfârșitul efectiv al elementului copil se va stabili după cum elementul copil are o durată sau un sfârșit explicit conform regulilor următoare:
  - Dacă un copil al unui element `par` are o durată explicită sau un sfârșit explicit, sfârșitul efectiv al elementului respectiv este stabilit ca și cum elementul ar avea un atribut `fill` cu valoarea `remove`.
  - Dacă un copil al unui element `par` nu are nici o durată explicită și nici un sfârșit explicit, sfârșitul efectiv al elementului respectiv este stabilit ca și cum elementul ar avea un atribut `fill` cu valoarea `freeze`.
- Sfârșitul efectiv al ultimului element copil al unui element `seq` este stabilit analog cu sfârșitul efectiv al oricărui element copil al unui element `par`.
- Sfârșitul efectiv al oricărui element copil al unui element `seq` care nu este terminal este stabilit astfel:
  - Dacă elementul copil are un atribut `fill` cu valoarea `freeze`, sfârșitul efectiv al elementului copil respectiv este egal cu începutul efectiv al următorului element. Cu alte cuvinte, ultima stare a elementului respectiv va rămâne pe ecran până când va începe următorul element, dacă durata sa, fie implicită, fie explicită, este mai mică decât diferența dintre începutul său și începutul următorului element.
  - Dacă elementul copil, care nu este terminal, al elementului `seq` are un atribut `fill` cu valoarea `remove`, sfârșitul efectiv al elementului copil respectiv este cel mai mic dintre începutul efectiv al următorului element și sfârșitul dorit al elementului.
  - Dacă un element copil, care nu este terminal, al unui element `seq` nu are un atribut `fill`, sfârșitul efectiv al elementului copil se va stabili după cum elementul copil are o durată sau un sfârșit explicit.

- Dacă un element copil, care nu este terminal, al unui element seq are o durată explicită sau un sfârșit explicit, sfârșitul efectiv al elementului respectiv este stabilit ca și cum elementul ar avea un atribut fill cu valoarea remove.
- Dacă un element copil, care nu este terminal, al unui element seq nu are nici o durată explicită și nici un sfârșit explicit, sfârșitul efectiv al elementului respectiv este stabilit ca și cum elementul ar avea un atribut fill cu valoarea freeze.

Toate acestea sunt un mod mai complicat de a spune că atributul fill cu valorile remove sau freeze va fi impus unui element care este conținut în altul, fie că vă place sau nu, deci ar trebui să vă gândiți la acest lucru de la început. Regulile sunt destul de complexe, deci ar fi mai înțelept să definiți explicit comportamentul fiecărui element decât să vă bazați pe comportamentul prestabilit, cu excepția cazului în care sunteți foarte sigur care este comportamentul prestabilit într-o anumită situație.

Elementul switch

Elementul switch îi permite autorului să selecteze din mai multe obiecte multimedia alternative, în funcție de teste care se pot realiza asupra variabilelor de sistem setate de utilizator sau de aptitudinile agentului utilizator. În toate cazurile, primul test satisfăcut este cel ale cărui elemente vor guverna comportamentul agentului utilizator. Dacă nu este satisfăcut nici un test, atunci nu se realizează nimic. Prezența oricărei variabile de sistem presupune un test, și de fapt un element switch cu o singură alegere, implicit dacă este prezentă orice variabilă de sistem. În acest caz, elementul părinte switch este opțional.
Elementul switch poate primi atribute id și title și poate să conțină următoarele elemente copil, care, la rândul lor, pot să primească unul sau mai multe dintre atributele variabile de sistem care implică un test:

- a
- animation
- audio
- img
- par
- ref
- seq
- switch
- text
- textstream
- video

În SMIL 1.0 sunt disponibile următoarele teste pentru atribute variabile de sistem:

- **system-bitrate** – Testează viteza conexiunii la rețea.
- **system-captions** – Testează dacă utilizatorul dorește afișarea legendelor elementelor.
- **system-language** – Testează limba stabilită de utilizator.
- **system-overdub-or-caption** – Testează dacă se va reda un conținut audio sau unul video.
- **system-required** – Testează prezența unui indicator stabilit de utilizator.
- **system-screen-size** – Testează dimensiunea ecranului agentului utilizator.
- **system-screen-depth** – Testează adâncimea de culoare a ecranului folosit de agentul utilizator.

Aceste teste pot fi combinate arbitrar, deci următorul cod switch va selecta una dintre cele cinci alternative: audio de rezoluție scăzută fie în franceză, fie în germană; audio de rezoluție ridicată fie în franceză, fie în germană și un fișier prestabilit, disponibil dacă cele patru combinații de teste eșuează dintr-un motiv sau altul.

```
<switch>
  <audio src="audiofile1" system-language="fr" system-bitrate="2400" />
  <audio src="audiofile2" system-language="de" system-bitrate="2400" />
  <audio src="audiofile3" system-language="fr" system-bitrate="4800" />
  <audio src="audiofile4" system-language="de" system-bitrate="4800" />
  <audio src="audiofile5" />
</switch>
```

Însă, în cazul în care doriți să fie executat un singur element în funcție de setarea variabilelor de sistem, puteți pune această instrucțiune in-line:

```
<par>
  <video src="videofile" />
  <audio src="audiofile" />
    <!-- Content for persons with visual disabilities -->
    <audio src="descriptiveaudiofile"
           system-captions="on"
           system-overdub-or-caption="overdub" />
    <!-- Content for persons with hearing disabilities -->
    <textstream src="closedcaptionfile"
           system-captions="on"
           system-overdub-or-caption="caption" />
</par>
```

Acest fragment redă un clip video însoțit de o secvență audio indiferent de setările utilizatorului. Dacă un utilizator dorește fie o secvență audio descriptivă, fie o legendă pe lângă programul de bază, va trebui să seteze variabila system-captions cu valoarea "on". Setarea variabilei system-overdub-or-caption va selecta apoi ceea ce se va reda, fie legenda, fie secvența audio descriptivă.

Atribute de test

Atributele pentru testarea sistemului par să declare la prima vedere ceea ce se va întâmpla într-o anumită situație. Cele două atribute care controlează legendele și înregistrările sonore par să se completeze reciproc, de exemplu, system-overdub-or-caption pare să aibă o legătură logică cu system-captions, dar de fapt nu sunt cu nimic mai înrudite decât orice alte două semafoare. După cum ați văzut în exemplul din secțiunea anterioară, acestea pot fi utilizate pentru a implementa caracteristicile respective, dar pot fi folosite la fel de bine și pentru orice altă caracteristică, dacă legendele și secvențele audio descriptive nu sunt necesare într-o anumită aplicație.
Unele dintre atribute apelează la comparații booleene între opțiuni care se exclud reciproc, iar altele compară sau chiar selectează dintr-o listă de opțiuni pentru a realiza comparații de subșiruri mai sofisticate.
În lista următoare sunt descrise în detaliu comportamentele fiecărei variabile de sistem:

- **system-bitrate** – Evaluează rata de transfer declarată de un agent utilizator față de rata de transfer cerută de un anumit element. Aceasta vă permite să încărcați fișiere mai voluminoase în cazul unei conexiuni de mare viteză și fișiere mai firave în cazul unor linii telefonice cu viteză redusă.

- **system-captions** – În funcție de setările on și off, se alege afișarea legendelor elementelor așa cum dorește utilizatorul. Deoarece există două atribute care se referă la legende, acestea pot fi folosite în combinații logice destul de sofisticate.
- **system-language** – În funcție de limbile disponibile într-o listă a agentului utilizator, este aleasă limba cea mai potrivită pentru afișare sau redare.
- **system-overdub-or-caption** – În funcție de setările overdub și caption, se alege redarea unei secvențe audio descriptive sau afișarea legendelor elementelor așa cum dorește utilizatorul. Deoarece există două atribute care se referă la legende, acestea pot fi folosite în combinații logice destul de sofisticate.
- **system-required** – Specifică numele unei extensii la funcționalitatea de bază a SMIL. Dacă agentul utilizator suportă această extensie, elementul care o cere este ales. Este posibil ca aici să apară de fapt o zonă de nume XML.
- **system-screen-size** – În funcție de dimensiunea ecranului raportată de agentul utilizator, se alege o dimensiune potrivită pentru regiune și pentru fișierele multimedia.
- **system-screen-depth** – În funcție de adâncimea de culoare raportată a agentului utilizator, se aleg fișierele multimedia cele mai potrivite pentru afișare.

Elemente de hiperlegături

SMIL are mare nevoie de facilități excelente pentru legare, deoarece în acest caz ar fi posibil să se adune conținutul multimedia de oriunde din Web și să fie integrat într-o producție multimedia continuă și uniformă. Însă tot ce este disponibil în momentul actual este un element a și un element anchor care se comportă ca o hartă client-side, dar pentru care sunt permise numai regiuni dreptunghiulare. SMIL Boston, succesorul lui SMIL 1.0 din următoarea generație planificată, nu pare să fie cu ceva mai bun în această privință, deși să sperăm că cei de la XLink vor termina în sfârșit și XLink va fi imediat încorporat în SMIL.

Elementul a

Acest element se comportă destul de asemănător cu elementul legătură din HTML, cu excepția faptului că SMIL adaugă un atribut show care controlează comportamentul documentului când este urmărită o legătură. Atributul show poate lua următoarele valori: replace (valoare prestabilită), care întrerupe prezentarea curentă și o înlocuiește cu resursa de la destinație; new, care creează un nou context de prezentare și lasă prezentarea originală să continue în paralel, și pause, care se comportă ca și new, doar că prezentarea originală este în pauză.

În afară de acestea elementul a mai poate primi următoarele atribute: id, href și title.

Elementul a poate să conțină următoarele elemente copil:

- animation
- audio
- img
- par
- ref
- seq
- switch
- text

- textstream
- video

Elementul a funcționează aproape exact la fel cu elementul a din HTML și poate să includă orice conținut multimedia în afară de un alt element a sau un element anchor:

```
<a href="URI"><video src="videofile" /></a>
```

Nu este de loc dificil.

Elementul anchor

Elementul anchor este într-o oarecare măsură similar cu o hartă imagine client-side, prin faptul că permite ca un obiect să fie separat în mod logic în părți. Deoarece este vorba de SMIL, care se bazează pe timp, puteți diviza un obiect nu numai în spațiu ci și în timp. Acest element este întotdeauna vid și pentru a avea efect trebuie să fie conținut într-un element multimedia.

Elementul anchor poate primi următoarele atribute:

- **begin** – Acest atribut specifică momentul de început al unui element. În combinație cu atributul end vă permite să tăiați clipul.
- **coords** – Definește o secțiune dreptunghiulară a unei imagini printr-o listă de patru valori separate prin virgule. Primele două valori precizează colțul din stânga-sus al secțiunii, iar ultimele două precizează colțul din dreapta-jos. Toate coordonatele sunt relative față de colțul din stânga-sus al obiectului și pot fi exprimate fie în pixeli (în mod prestabilit), fie ca procent.
- **end** – Acest atribut specifică sfârșitul unui element. În combinație cu atributul begin vă permite să tăiați clipul.
- **id** – Un identificator unic.
- **show** – Atributul show poate să ia următoarele valori: replace (valoare prestabilită), care întrerupe prezentarea curentă și o înlocuiește cu resursa de la destinație; new, care creează un nou context de prezentare și lasă prezentarea originală să continue în paralel, așa cum apariția unei noi ferestre de browser nu afectează cu nimic fereastra de dedesubt, și pause, care se comportă ca și new, doar că prezentarea originală este în pauză.
- **skip-content** – Acest atribut boolean permite procesoarelor SMIL 1.0 să ignore conținutul elementului anchor dacă o specificare ulterioară îl modifică de la un element vid la unul cu conținut.
- **title** – Un titlu pentru element, care poate sau nu să fie redat, în funcție de agentul utilizator și de cerințele particulare ale utilizatorului.

Prezentări fotografice

O prezentare fotografică este cel mai simplu lucru care poate fi realizat cu ajutorul SMIL. Un simplu element secvență seq este folosit pentru a controla înșiruirea de clișee, fiecare dintre acestea având o durată de cinci secunde.

Mai întâi se definește o regiune de vizualizare pentru prezentarea fotografică, folosind elementul layout și două elemente copil, root-layout și region în antetul SMIL. Apoi se definește secvența de clișee în corpul SMIL indicând printr-un atribut region atașat fiecărui element multimedia id-ul singurei regiuni definite:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 1.0//EN"
          "http://www.w3.org/TR/REC-smil/SMIL10.dtd">
  <smil>
    <head>
      <layout>
        <root-layout id="Slideshow" title="ALS Slideshow Window"
           width="400" height="400" />
        <region id="slides" title="ALS Slide"
          width="400" height="300" fit="meet" />
      </layout>
    </head>
    <body>
      <seq dur="indefinite" repeat="indefinite">
        <img region="slides" src="29-1.gif" dur="5s" />
        <img region="slides" src="29-2.gif" dur="5s" />
        <img region="slides" src="29-3.gif" dur="5s" />
        <img region="slides" src="29-4.gif" dur="5s" />
        <img region="slides" src="29-5.gif" dur="5s" />
        <img region="slides" src="29-6.gif" dur="5s" />
        <img region="slides" src="29-7.gif" dur="5s" />
      </seq>
    </body>
  </smil>
```

Acest mic document SMIL redă o prezentare fotografică simplă care se repetă la infinit. O puteţi încerca afişând imagini din colecţia dumneavoastră de fotografii cu rude îndepărtate, dacă aveţi nişte oaspeţi plictisitori de care doriţi să scăpaţi într-un mod politicos. Dar acesta este şi un bun punct de plecare pentru dezvoltarea unei prezentări. Veţi vedea în secţiunea următoare cum puteţi realiza acest lucru. În acest document, declaraţia `xml` şi referinţa `DTD` sunt opţionale, deşi este o idee bună să vă obişnuiţi să declaraţi ambele elemente în toate documentele XML, fie că le veţi utiliza într-un browser care validează, fie că nu.

Prezentări multimedia

Desigur, dacă tot ce aţi putea face în SMIL ar fi simple prezentări fotografice, atunci nu prea ar avea rost. Puterea reală a limbajului apare atunci când sincronizaţi surse audio şi alte surse multimedia cu prezentarea fotografică sau când adăugaţi imagini la o producţie audio şi video, sau mixaţi sfârşitul unei piese muzicale cu începutul estompat al altei piese muzicale. Pentru a adăuga muzică sau naraţiune la prezentarea fotografică dată ca exemplu, veţi proceda astfel:

Mai întâi includeţi fiecare element `img` într-un element paralel `par` exterior, pentru a vă asigura că sunt sincronizate cu orice conţinut cu care veţi dori să fie redate simultan. Apoi adăugaţi elementele necesare pentru includerea noului material multimedia. Deocamdată să adăugăm câte un fragment audio la fiecare clişeu:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 1.0//EN"
          "http://www.w3.org/TR/REC-smil/SMIL10.dtd">
  <smil>
    <head>
      <layout>
```



```
        <root-layout id="Slideshow" title="ALS Slideshow Window"
           width="400" height="400" />
        <region id="slides" title="ALS Slide"
          width="400" height="300" fit="meet" />
        <region id="audio" />
      </layout>
    </head>
    <body>
      <seq dur="indefinite" repeat="indefinite">
        <par>
          <img src="29-1.gif" dur="5s" />
          <audio src="29-1.wav" />
        </par>
        <par>
          <img src="29-2.gif" dur="5s" />
          <audio src="29-2.wav" />
        </par>
        <par>
          <img src="29-3.gif" dur="5s" />
          <audio src="29-3.wav" />
        </par>
        <par>
          <img src="29-4.gif" dur="5s" />
          <audio src="29-4.wav" />
        </par>
        <par>
          <img src="29-5.gif" dur="5s" />
          <audio src="29-5.wav" />
        </par>
        <par>
          <img src="29-6.gif" dur="5s" />
          <audio src="29-6.wav" />
        </par>
        <par>
          <img src="29-7.gif" dur="5s" />
          <audio src="29-7.wav" />
        </par>
      </seq>
    </body>
  </smil>
```

Notă Fişierele audio pot fi fişiere RealSystem G2, sau orice alt format audio suportat de programul dumneavoastră de redare SMIL.

Dacă pare simplu numai citind codul, aşa şi este. Aproape orice prezentare multimedia este o combinaţie de lucruri care se petrec secvenţial şi de lucruri care se petrec în paralel (mai mult sau mai puţin). Elementele `seq` şi `par` controlează aceste niveluri temporale. În Figura 23.1 de mai sus aţi văzut o prezentare fotografică asemănătoare, creată manual şi redată de motorul de redare de bază, RealPlayer 7.

După ce aţi început, poate vă veţi gândi să folosiţi un editor special pentru a controla asamblarea tuturor componentelor. Figura 23.2 vă prezintă editorul GRiNS de la Oratrix, `http://www.oratrix.com`, în plină acţiune, editând fişierul anterior, deşi are unele

probleme, cum se întâmplă cu mai toate editoarele.
Observați că se ascunde complet codul SMIL efectiv. Puteți să creați legături și să modificați timpii, și chiar să modificați numele de `id` și orice alte atribute, numai prin folosirea mouse-ului pe o interfață GUI. După ce ați terminat de editat, puteți salva fișierul și documentul SMIL este gata de redare. Totuși, ca în cazul multor furnizori patentați, se indică un DTD patentat, nu DTD-ul standard aflat la W3C și se folosesc câteva atribute care nu apar în standard. Deci va trebui să mai editați manual noul și grozavul dumneavoastră cod SMIL eliminând particularitățile, pentru a fi cu adevărat valid.

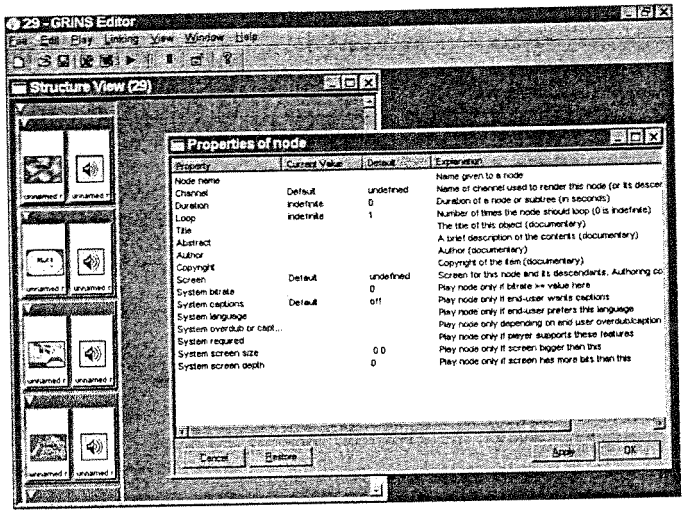


FIGURA 23.2
Editorul GRiNS în timp ce editează un fișier SMIL simplu.

Tipuri multimedia

SMIL folosește tipurile multimedia standard Internet, ca regulă generală, ori de câte ori este necesar un tip. *MIME*, sau *Multipurpose Internet Mail Extensions*, cunoscut și ca *Internet Media Types* (*IMT*), este definit în RFC 1521. Cele șapte categorii de bază sunt `text` (text), `multipart` (mai multe părți), `message` (mesaj), `image` (imagine), `audio` (audio), `video` (video) și `application` (aplicație), care acoperă majoritatea formatelor multimedia patentate. Tipul de date pentru SMIL în sine este `text/smil-basic-layout`.
Tabelul 23.1 vă prezintă o listă scurtă de tipuri multimedia folosite de obicei în prezentări, împreună cu extensiile folosite pentru descrierea fișierelor într-un mediu moștenit din DOS.

Tabelul 23.1 Tipuri multimedia generale

Tip multimedia	Extensia generală a fișierului
`application/octet-stream`	`bin`
`application/oda`	`oda`
`application/pdf`	`pdf`
`application/postscript`	`ai, eps, ps`
`audio/32kadpcm audio/basic`	`au, snd`
`audio/x-aiff`	`aif, aiff, aifc`
`audio/x-wav`	`wav`
`image/cgm`	`cgm`
`image/gif`	`gif`
`image/jpeg`	`jpeg, jpg, jpe`
`image/naplps`	`naplps`
`image/tiff`	`tiff, tif`
`image/x-portable-bitmap`	`pbm`
`image/x-portable-graymap`	`pgm`
`image/x-portable-pixmap`	`ppm`
`image/x-rgb`	`rgb`
`image/x-xbitmap`	`xbm`
`image/x-xpixmap`	`xpm`
`text/html`	`html, htm`
`text/plain`	`txt`
`text/richtext`	`rtx`
`video/mpeg`	`mpeg, mpg, mpe`
`video/quicktime`	`qt, mov`
`video/x-sgi-movie`	`movie`

Audio

Pentru mulți dintre noi componenta audio este importantă, motiv pentru care puțini își cumpără o mașină fără radiocasetofon. Deci o mare parte a lui „multi" din multimedia este adăugarea sunetului. Calculatoarele dispun de multă vreme de o grafică performantă, în schimb sunetul de calitate a devenit disponibil doar recent. Înainte, tehnologia consta dintr-un difuzor care era pornit și oprit într-un mod destul de primitiv, producând astfel unde dreptunghiulare, iar sunetul era, cinstit vorbind, jalnic.
Calculatoarele de astăzi conțin plăci de sunet capabile de performanțe care acum câțiva ani puteau fi obținute numai de sistemele audio de înaltă fidelitate. Plăcile de sunet au trecut de la stadiul de jucării scumpe la o necesitate în mediul de afaceri de astăzi. Având o placă de sunet și un microfon mic, puteți vorbi acasă prin Internet. Cunosc o femeie care a oferit părinților săi un calculator destul de performant și a reușit să îl plătească în interval de un an numai din economiile făcute la telefon.
Și dacă adăugați un sistem de boxe performant – cum ar fi sistemul de sunet surround cu cinci boxe pe care îl am eu conectat, împreună cu un singur difuzor de joase, care poate să zguduie ferestrele – și o placă video, aveți în față varianta la scară redusă a unui sistem de sunet care costa zeci de mii de dolari în urmă cu zece ani. Cu un ecran mare, sau conectând calculatorul la un monitor de proiecție închiriat, puteți să prezentați la nivel profesional propriile dumneavoastră producții, incluzând muzică și toate celelalte ornamente, pe un calculator care costă cam o mie de dolari. Desigur, cu cât aveți mai multă memorie și o viteză mai mare, cu atât este mai bine pentru sistemele folosite în proiectarea grafică și video, deci nu este greu să cheltuiți mai mult.

În schimb, în 1988 am specificat și am procurat pentru o mare corporație un sistem de sunet care includea o placă video de 10000$, un sistem de editare video de 60000$ și sisteme anexe care au dus prețul la un număr cu șase cifre. Era la acel moment cel mai bun de pe piață, dar în ziua de azi puteți face mult mai multe și mult mai rapid cu Video Toaster pe un sistem Windows NT care costă mai puțin de 15000$, dar cu o memorie de un gigaoctet, un procesor Pentium III la 600 MHz și multe alte componente. Dacă aruncați o privire la orice film actual cu efecte speciale, veți vedea ce a făcut pentru industria video puterea de procesare (relativ) ieftină.

Toate acestea spun că SMIL nu putea să apară într-un moment mai favorabil. Chiar atunci când hardware-ul ieftin face posibil ca un amator talentat să producă filme și multimedia care pot rivaliza cu cele realizate de mari studiouri, un standard multimedia open-source devine limbajul comun pentru prezentările de pe Web.

Următoarele secțiuni vă prezintă alte câteva limbaje multimedia XML disponibile pentru prezentările audio.

VoxML Mai există și alte limbaje de marcare care se adresează unor segmente de utilizatori foarte specializate, cum ar fi VoxML, care este proiectat pentru a reda text în vorbire și voce preînregistrată într-un sistem interactiv de răspuns vocal. Deși împărtășește o moștenire XML, vocabularul elementelor este proiectat special pentru mediul de răspunsuri vocale. Iată un exemplu de cod, ca să știți măcar cum arată:

```
<?xml version="1.0"?>
  <DIALOG>
    <STEP NAME="initialize">
      <PROMPT>Hello. This is a voice response system.</PROMPT>
    </STEP>
  </DIALOG>
```

Nu putem să descriem în amănunt această tehnologie în spațiul acestei secțiuni, dar pe situl VoxML al Motorola, la `http://www.voxml.com/`, găsiți un set de dezvoltare pe care îl puteți descărca dacă sunteți interesat să explorați mai departe subiectul.

VoiceXML IBM, Lucent Technologies, Motorola și AT&T au o tehnologie concurentă, VoiceXML (`http://www.voicexml.org/`), care folosește un exemplu foarte asemănător cu următorul pentru a demonstra cât de comodă este această tehnologie:

```
<vxml>
  <form>
    <field name="drinkrequest">
      <prompt>Would you like coffee, tea, or milk?</prompt>
      <grammar src="drinkrequest.gram"/>
    </field>
    <block>
      <goto next="http://www.drink.example/drink2.asp"
        submit="drinkrequest" method="get"/>
    </block>
  </form>
</vxml>
```

În VoiceXML, un câmp este un câmp de intrare. Utilizatorul trebuie să furnizeze o valoare „legitimă" pentru câmpul respectiv, pronunțând „corect" un cuvânt dintr-o listă de câteva cuvinte pe care calculatorul le „înțelege" înainte de a trece la următorul element din formular. Logica de ciclare este furnizată intern de către procesor. Deși totul poate să meargă după plan, dacă utilizatorii nu se depărtează de un utilizator complezent „ideal", un accent sau o particularitate de pronunție va împiedica complet înțelegerea, iar încercarea de a trece peste granițele rigide impuse de programator este sortită eșecului. Un exemplu realist de interacțiune între un calculator și un om care vorbește cu glas tare și nu tastează, ar putea fi:

```
Computer: Would you like coffee, tea, or milk?
Human (in thick Scottish accent): Orange juice.
Computer: I did not understand what you said.
Computer: Would you like coffee, tea, or milk?
Human: Chai with nutmeg.
Computer: I did not understand what you said.
Computer: Would you like coffee, tea, or milk?
Human: I give up! Tea!
Computer: I did not understand what you said.
Computer: Would you like coffee, tea, or milk?
Human: Tea!
Computer: I did not understand what you said.
Computer: Would you like coffee, tea, or milk?
Human: Coffee!
Computer: Thank you.
```

adică

```
Calculator: Doriti cafea, ceai sau lapte?
Persoana (cu un puternic accent scotian): Suc de portocale.
Calculator: Nu înteleg ce spuneti.
Calculator: Doriti cafea, ceai sau lapte?
Persoana: Ceai cu nucsoara.
Calculator: Nu înteleg ce spuneti.
Calculator: Doriti cafea, ceai sau lapte?
Persoana: Ma dau batut! Ceai!
Calculator: Nu înteleg ce spuneti.
Calculator: Doriti cafea, ceai sau lapte?
Persoana: Ceai!
Calculator: Nu înteleg ce spuneti.
Calculator: Doriti cafea, ceai sau lapte?
Persoana: Cafea!
Calculator: Multumesc.
```

În momentul de față, sistemele de răspuns vocal sunt foarte limitate, în ciuda insistențelor entuziaștilor acestor sisteme, care susțin că sunt gata pentru a intra în prim plan. Eu personal nu cred că am avut vreodată o conversație satisfăcătoare cu un astfel de sistem. Având în vedere stadiul actual, în afară de cazul în care întreaga discuție ar putea fi limitată drastic iar utilizatorii ar ști la ce să se aștepte și ar putea să se stăpânească și să gândească asemenea unui calculator, aceste sisteme tind să dezamăgească și chiar să irite. Tehnologia este foarte departe de Star Trek, sau chiar de HAL în acest moment. Dar cum vitezele calculatoarelor se apropie de 1000 MHz și chiar mai mult, iar spațiul în memoria principală ajunge în domeniul teraocteților, ne putem aștepta să vedem în următorii zece ani sisteme care să se apropie de capacitatea de conversație a unui cimpanzeu destul de inteligent.

Riscând o estimare, cred că HAL și poate chiar Star Trek vor fi posibile înainte de jumătatea secolului. Dar nu acum.

Când proiectați un sistem de răspuns vocal interactiv (IVR – interactive voice response), asigurați-vă întotdeauna că țineți cont de faptul că unele persoane nu se vor putea înțelege cu calculatorul, oricât de mult s-ar strădui. Poate că nu au un telefon prea bun. Orice sistem IVR trebuie să aibă câțiva operatori pregătiți pentru a se ocupa de persoanele care nu se descurcă, sau o modalitate clară de a intra în contact cu aceștia fără proceduri complicate și vorbe fără sens. Sugerez eticheta prestabilită „Operator" și conectarea automată la un operator după o perioadă scurtă de liniște sau de zgomote „ciudate". Ar putea fi de exemplu, cineva care încearcă să comunice cu un TDD.

Întreruperea brutală a legăturii, așa cum se întâmplă în cazul multor sisteme IVR, este o impolitețe de neiertat și chiar o încălcare potențială a legilor referitoare la persoanele cu handicap, ca să nu mai spunem că este un gest demn de dispreț.

Video

Pe Web există mai multe formate video, RealSystem G2 și QuickTime fiind cele mai răspândite, după cum o demonstrează utilizările efective. Ambele sisteme sunt la ora actuală mai mult sau mai puțin compatibile cu SMIL. Facilitățile existente în SMIL vă permit să selectați videoclipuri din fișiere mai mari cu o precizie de o sutime de cadru, dacă în codul fișierului există informații SMPTE. În caz contrar, informațiile NPT pot furniza date de poziționare mai puțin precise.

Text

Textul nu are în mod intrinsec o valoare de timp asociată, deci în afară de afișare și de oprirea afișării când ați obosit să vă uitați la ecran, nu se poate face mare lucru cu el. De obicei, textul este folosit la titluri și la legende, dar nu există nici un motiv pentru care să nu puteți realiza o prezentare numai din text, dacă vă puneți în gând.

Coordonarea cu instrumentele multimedia existente (ShockWave)

Dreamweaver de la Macromedia, la `http://www.macromedia.com/`, este un mediu de creare compatibil cu programul de redare RealPlayer 7 de la RealNetworks, la `http://www.real.com/`. RealSlideshow Basic este disponibil gratuit pe acest sit și este un editor multimedia simplu care vă permite să creați ușor documente SMIL și, de asemenea, să utilizați limbajul nativ al lui RealPlayer 7, permițându-vă un control mai strict asupra prezentărilor video. Puteți crea documente multimedia bazate pe HTML folosind tipurile multimedia din RealPlayer 7, care cuprind RealAudio, RealVideo, RealFlash, RealPix și RealText. Dar veți avea nevoie de programul de redare RealPlayer 7 sau poate de un echivalent Microsoft ca plug-in, altfel nu veți putea reda aceste formate multimedia.

În mod interesant, RealPlayer 7 suportă și SMIL, deci același plug-in poate fi folosit pentru mai multe formate, inclusiv pentru cele complet deschise.

Narațiunea și legendele

Caracteristica de bază a SMIL este accesibilitatea, deși autorul va trebui să muncească puțin pentru a obține această caracteristică pentru o audiență cât mai mare. Există două formate alternative de bază disponibile pentru persoanele cu handicap de percepție: narațiunea descriptivă și legendele.

Narațiunea poate fi împletită cu secvența audio astfel încât toată lumea să audă același lucru, dar pentru a obține rezultatele cele mai bune este mai bine să furnizați o a doua secvență audio dedicată narațiunii. Unele persoane vor distinge cu greu cuvintele dacă există un fundal sonor, deci dacă asigurați canale separate permiteți persoanelor să aleagă secvențele care vor fi redate. Acest al doilea canal este de asemenea o modalitate excelentă de a asigura capabilitatea pentru a doua limbă.

Legendele permit persoanelor cu handicap auditiv să citească descrierea unei informații audio care este redată în documentul SMIL. Legendele pot fi folosite și ca intrări pentru un cititor auditiv de ecran, deci sunt o altă modalitate de a furniza conținut pentru persoanele cu handicap de vedere. Deci în continuarea exemplului, la prezentarea fotografică, ar trebui să adăugați atât titluri cât și legende.

Dacă tot faceți asta, definiți și referiți regiuni particulare în care doriți să ajungă fiecare componentă. Până acum am lăsat totul la întâmplare, deoarece oricum trebuia ca imaginile să ajungă în același loc. Deci, acest fișier va fi ceva mai mare și veți vedea de ce folosirea unui editor specializat poate la un anumit moment să fie o idee bună.

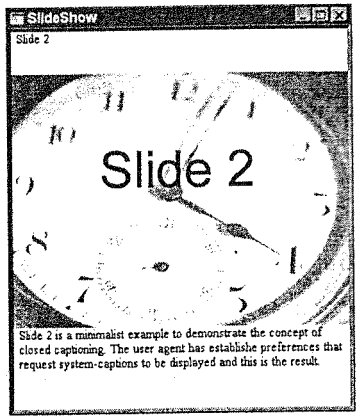
Observați că în această versiune au fost adăugate pentru majoritatea elementelor etichetele `id`, `title` și `alt`. Specificația actuală încurajează utilizarea etichetelor `title` și `alt` pentru toate elementele care le suportă. În `switch` există doar un singur element, deoarece legendele nu trebuie afișate decât dacă cineva le cere. Totuși, ați putea folosi elementul pentru a alege din mai multe opțiuni, dacă doriți. De exemplu, ați putea furniza versiuni ale legendelor în spaniolă și engleză:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 1.0//EN"
          "http://www.w3.org/TR/REC-smil/SMIL10.dtd">
<smil>
  <head>
    <meta name="title" content="Slideshow"/>
    <layout>
      <root-layout id="Slideshow" width="400" height="450" />
      <region id="slides" title="Slides" width="400" height="300" top="50" />
      <region id="audio" title="Audio" />
      <region id="titles" title="Titles" width="400" height="50" />
      <region id="captions" title="Closed Captions" width="400" height="50"
top="350" />
    </layout>
  </head>
  <body>
    <seq region="Slideshow" dur="indefinite" repeat="indefinite">
      <par id="Slide-1" title="One">
        <img region="slides" src="29-1.gif" dur="5s" alt="Slide 1" />
        <audio region="audio" src="29-1.wav" alt="Slide 1 Audio" />
        <text region="titles" src="29-1.txt" type="text/plain" />
        <switch title="Switch 1">
          <text region="captions" src="29-1cc.txt" type="text/plain"
            system-captions="on" />
        </switch>
      </par>
      <par id="Slide-2" title="Two">
        <img region="slides" src="29-2.gif" dur="5s" alt="Slide 2" />
        <audio region="audio" src="29-2.wav" alt="Slide 2 Audio" />
```

```
    <text region="titles" src="29-2.txt" type="text/plain" />
    <switch title="Switch 2">
      <text region="captions" src="29-2cc.txt" type="text/plain"
        system-captions="on" />
    </switch>
  </par>
  <par id="Slide-3" title="Three">
    <img region="slides" src="29-3.gif" dur="5s" alt="Slide 3" />
    <audio region="audio" src="29-3.wav" alt="Slide 3 Audio" />
    <text region="titles" src="29-3.txt" type="text/plain" />
    <switch title="Switch 3">
      <text region="captions" src="29-3cc.txt" type="text/plain"
        system-captions="on" />
    </switch>
  </par>
  <par id="Slide-4" title="Four">
    <img region="slides" src="29-4.gif" dur="5s" alt="Slide 4" />
    <audio region="audio" src="29-4.wav" alt="Slide 4 Audio" />
    <text region="titles" src="29-4.txt" type="text/plain" />
    <switch title="Switch 4">
      <text region="captions" src="29-4cc.txt" type="text/plain"
        system-captions="on" />
    <text region="titles" src="29-1.txt" type="text/plain" />
    </switch>
  </par>
  <par id="Slide-5" title="Five">
    <img region="slides" src="29-5.gif" dur="5s" alt="Slide 5" />
    <audio region="audio" src="29-5.wav" alt="Slide 5 Audio" />
    <text region="titles" src="29-5.txt" type="text/plain" />
    <switch title="Switch 5">
      <text region="captions" src="29-5cc.txt" type="text/plain"
        system-captions="on" />
    </switch>
  </par>
  <par id="Slide-6" title="Six">
    <img region="slides" src="29-6.gif" dur="5s" alt="Slide 6" />
    <audio region="audio" src="29-6.wav" alt="Slide 6 Audio" />
    <text region="titles" src="29-6.txt" type="text/plain" />
    <switch title="Switch 6">
      <text region="captions" src="29-6cc.txt" type="text/plain"
        system-captions="on" />
    </switch>
  </par>
  <par id="Slide-7" title="Seven">
    <img region="slides" src="29-7.gif" dur="5s" alt="Slide 7" />
    <audio region="audio" src="29-7.wav" alt="Slide 7 Audio" />
    <text region="titles" src="29-7.txt" type="text/plain"/>
    <switch title="Switch 7">
      <text region="captions" src="29-7cc.txt" type="text/plain"
        system-captions="on" />
    </switch>
  </par>
 </seq>
</body>
</smil>
```

Figura 23.3 vă prezintă editorul GRiNS al Oratrix de la `http://www.oratrix.com/` redând această prezentare fotografică.

FIGURA 23.3
Programul de editare/redare GRiNS vă arată în întregime prezentarea fotografică împreună cu legendele asociate.



Secvenţele audio descriptive

O secvenţă audio descriptivă este de obicei o a doua secvenţă care descrie ceea ce se întâmplă din punct de vedere vizual, în timp ce secvenţa principală conţine porţiunea audio prestabilită a prezentării. Există mai multe opţiuni la care trebuie să vă gândiţi atunci când adăugaţi secvenţe audio descriptive: doriţi să intercalaţi secvenţele audio descriptive între dialoguri – cum se întâmplă adesea în producţiile de televiziune – sau ca voce peste un fundal sonor, sau ar trebui să aşteptaţi sfârşitul secvenţei audio prestabilite şi apoi să introduceţi naraţiunea? În acest caz, este mai logic să aşteptaţi încheierea secvenţei audio prestabilite şi să adăugaţi apoi descrierea.

Bineînţeles, pentru a adăuga secvenţe audio descriptive la prezentarea dumneavoastră multimedia, aveţi nevoie de un nou element `switch` pentru a controla noul comportament şi trebuie să folosiţi atributul `system-overdub-or-caption="overdub"` pentru a alege în funcţie de preferinţele utilizatorului dacă se va reda sau nu secvenţa audio descriptivă. De asemenea, vom folosi şi un nou concept – utilizarea clipurilor dintr-o singură sursă audio care se presupune că include în codul său un tact SMPTE (în stil video) care ar putea fi adăugat de o staţie de editare video profesională. Atributele `clip-begin` şi `clip-end` primesc o valoare unică în care este inclus şi un al doilea semn de egalitate. Nu îi daţi atenţie. Aceste elemente sunt ilustrate în următorul exemplu de cod:

Sugestie de la LeeAnne.com Words to weave by

Nu este obligatoriu să puneţi un element `switch` în jurul unui singur element care va fi redat condiţionat. Însă dacă lăsaţi afară elementul `switch`, logica este foarte greu de observat, deci pare să merite să vă străduiţi să subliniaţi faptul că s-a ajuns la un punct în care se cere o decizie condiţionată.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE smil PUBLIC "-//W3C//DTD SMIL 1.0//EN"
        "http://www.w3.org/TR/REC-smil/SMIL10.dtd">
<smil>
  <head>
```

```
    <meta name="title" content="SlideShow"/>
    <layout>
      <root-layout id="SlideShow" width="400" height="450" />
      <region id="slides" title="Slides" width="400" height="300" top="50" />
      <region id="audio" title="Audio" />
      <region id="audio2" title="Descriptive Audio" />
      <region id="titles" title="Titles" width="400" height="50" />
      <region id="captions" title="Closed Captions" width="400" height="50"
top="350" />
    </layout>
  </head>
  <body>
    <seq id="Slide-Show" dur="indefinite" repeat="indefinite">
      <par id="Slide-1" title="One">
        <img region="slides" src="29-1.gif" dur="5s" alt="Slide 1" />
        <seq id="Slide-1DA" title="One DA">
          <audio region="audio" src="29-1.wav" alt="Slide 1 Audio" />
          <switch title="Switch 1 DA">
            <audio region="audio2" src="29-1-DA.au" alt="Slide 1 Audio DA"
              clip-begin="smpte=22:11:02.54" clip-end="smpte=22:16:21.33"
              system-overdub-or-caption="overdub" />
          </switch>
        </seq>
        <text region="titles" src="29-1.txt" type="text/plain" />
        <switch title="Switch 1 CC">
          <text region="captions" src="29-1cc.txt" type="text/plain"
            system-captions="on" />
        </switch>
      </par>
      <par id="Slide-2" title="Two">
        <img region="slides" src="29-2.gif" dur="5s" alt="Slide 2" />
        <seq id="Slide-2DA" title="Two DA">
          <audio region="audio" src="29-2.wav" alt="Slide 2 Audio" />
          <switch title="Switch 2 DA">
            <audio region="audio2" src="29-1-DA.au" alt="Slide 2 Audio DA"
              clip-begin="smpte=22:34:11.01" clip-end="smpte=22:39:24.57"
              system-overdub-or-caption="overdub" />
          </switch>
        </seq>
        <text region="titles" src="29-2.txt" type="text/plain" />
        <switch title="Switch 2 CC">
          <text region="captions" src="29-2cc.txt" type="text/plain"
            system-captions="on" />
        </switch>
      </par>
      <par id="Slide-3" title="Three">
        <img region="slides" src="29-3.gif" dur="5s" alt="Slide 3" />
        <seq id="Slide-3DA" title="Three DA">
          <audio region="audio" src="29-3.wav" alt="Slide 3 Audio" />
          <switch title="Switch 3 DA">
            <audio region="audio2" src="29-1-DA.au" alt="Slide 3 Audio DA"
              clip-begin="smpte=24:21:42.34" clip-end="smpte=24:26:28.11"
              system-overdub-or-caption="overdub" />
          </switch>
        </seq>
        <text region="titles" src="29-3.txt" type="text/plain" />
        <switch title="Switch 3 CC">
          <text region="captions" src="29-3cc.txt" type="text/plain"
            system-captions="on" />
        </switch>
      </par>
      <par id="Slide-4" title="Four">
        <img region="slides" src="29-4.gif" dur="5s" alt="Slide 4" />
        <seq id="Slide-4DA" title="Four DA">
          <audio region="audio" src="29-4.wav" alt="Slide 4 Audio" />
          <switch title="Switch 4 DA">
            <audio region="audio2" src="29-1-DA.au" alt="Slide 4 Audio DA"
              clip-begin="smpte=11:31:28.13" clip-end="smpte=11:36:22.21"
              system-overdub-or-caption="overdub" />
          </switch>
        </seq>
        <text region="titles" src="29-4.txt" type="text/plain" />
        <switch title="Switch 4 CC">
          <text region="captions" src="29-4cc.txt" type="text/plain"
            system-captions="on" />
        <text region="titles" src="29-1.txt" type="text/plain" />
        </switch>
      </par>
      <par id="Slide-5" title="Five">
        <img region="slides" src="29-5.gif" dur="5s" alt="Slide 5" />
        <seq id="Slide-5DA" title="Five DA">
          <audio region="audio" src="29-5.wav" alt="Slide 5 Audio" />
          <switch title="Switch 5 DA">
            <audio region="audio2" src="29-1-DA.au" alt="Slide 5 Audio DA"
              clip-begin="smpte=24:21:28.11" clip-end="smpte=24:26:28.11"
              system-overdub-or-caption="overdub" />
          </switch>
        </seq>
        <text region="titles" src="29-5.txt" type="text/plain" />
        <switch title="Switch 5 CC">
          <text region="captions" src="29-5cc.txt" type="text/plain"
            system-captions="on" />
        </switch>
      </par>
      <par id="Slide-6" title="Six">
        <img region="slides" src="29-6.gif" dur="5s" alt="Slide 6" />
        <seq id="Slide-6DA" title="Six DA">
          <audio region="audio" src="29-6.wav" alt="Slide 6 Audio" />
          <switch title="Switch 5 DA">
            <audio region="audio2" src="29-1-DA.au" alt="Slide 6 Audio DA"
              clip-begin="smpte=33:20" clip-end="smpte=39:25"
              system-overdub-or-caption="overdub" />
          </switch>
        </seq>
        <text region="titles" src="29-6.txt" type="text/plain" />
        <switch title="Switch 6 CC">
          <text region="captions" src="29-6cc.txt" type="text/plain"
```

```
          clip-begin="smpte=12:33:20" clip-end="smpte=12:33:25"
          system-captions="on" />
      </switch>
    </par>
    <par id="Slide-7" title="Seven">
      <img region="slides" src="29-7.gif" dur="5s" alt="Slide 7" />
      <seq id="Slide-7DA" title="Seven DA">
        <audio region="audio" src="29-7.wav" alt="Slide 7 Audio" />
        <switch title="Switch 5 DA">
          <audio region="audio2" src="29-1-DA.au" alt="Slide 7 Audio DA"
            clip-begin="smpte=28:21" clip-end="smpte=34:35"
           system-overdub-or-caption="overdub" />
        </switch>
      </seq>
      <text region="titles" src="29-7.txt" type="text/plain"/>
      <switch title="Switch 7 CC">
        <text region="captions" src="29-7cc.txt" type="text/plain"
          system-captions="on" />
      </switch>
    </par>
  </seq>
 </body>
</smil>
```

Observați că acest document, care deja este destul de complex, a fost creat într-o perioadă de timp, pornind de la o simplă prezentare fotografică și trecând la caracteristici din ce în ce mai multe și la o complexitate mai mare. Aceasta este o metodă excelentă de dezvoltare. Dacă creați rapid ceva care realizează o mare parte din operații, veți fi capabil să adăugați treptat funcționalități, fără să săriți direct la chestiunile complicate. Dezvoltarea treptată vă permite să vă opriți și să vă gândiți la toate și să prindeți greșelile una câte una și nu toate odată.

Problema accesibilității și a calculatoarelor portabile

Facilitățile actuale pentru accesibilitate și pentru calculatoarele portabile sunt destul de primitive, iar succesorul din generația următoare a lui SMIL 1.0 – care este încă în dezvoltare și poartă numele de cod „Boston" – va adăuga fără îndoială o mulțime de caracteristici pentru a suporta ambele facilități atunci când schița de lucru va fi aprobată și publicată. Există însă câteva elemente și atribute care se adresează direct acestor probleme. Prin utilizarea atributelor de test puteți croi afișajul în funcție de preferințele utilizatorului și de posibilitățile dispozitivului de ieșire:

- **`system-bitrate`** – Evaluează rata de transfer declarată de un agent utilizator față de rata de transfer cerută de un anumit element. Aceasta vă permite să încărcați fișiere mai voluminoase în cazul unei conexiuni de mare viteză și fișiere mai mici în cazul unor linii telefonice cu viteză redusă.
- **`system-captions`** – În funcție de setările `on` și `off` se alege afișarea legendelor elementelor așa cum dorește utilizatorul. Deoarece există două atribute care se referă la legende, acestea pot fi folosite în combinații logice destul de sofisticate.
- **`system-language`** – În funcție de limbile disponibile dintr-o listă a agentului utilizator este aleasă limba cea mai potrivită pentru afișare sau redare.

- **`system-overdub-or-caption`** – În funcție de setările `overdub` și `caption` se alege redarea unei secvențe audio descriptive sau afișarea legendelor elementelor așa cum dorește utilizatorul. Deoarece există două atribute care se referă la legende, acestea pot fi folosite în combinații logice destul de sofisticate.
- **`system-required`** – Specifică numele unei extensii față de funcționalitatea de bază a SMIL. Dacă agentul utilizator suportă această extensie, elementul care o cere este ales. Este posibil ca aici să apară de fapt o zonă de nume XML.
- **`system-screen-size`** – În funcție de dimensiunea ecranului raportată de agentul utilizator, se alege o dimensiune potrivită pentru regiune și pentru fișierele multimedia.
- **`system-screen-depth`** – În funcție de adâncimea de culoare raportată a agentului utilizator se aleg fișierele multimedia cele mai potrivite pentru afișare.

Dispozitivele portabile devin din ce în ce mai mici și vor avea și în viitorul apropiat o adâncime de culoare mai mică decât calculatorul de acasă. De asemenea, de obicei sunt conectate la Internet prin legături mai lente. Asta înseamnă că probabil nu se vor putea folosi aceleași fișiere în ambele situații. Figura 23.4 vă prezintă câteva dispozitive de comunicație mobile care au capacități Web, de la un ecran în mod text pe 11 linii în stânga până la un afișaj în nuanțe de gri semi-VGA în partea de jos și un ecran VGA color în dreapta sus. Aceste dispozitive sunt furnizate de NeoPoint (primul) și Psion (celelalte), dar dispozitive similare apar peste tot.

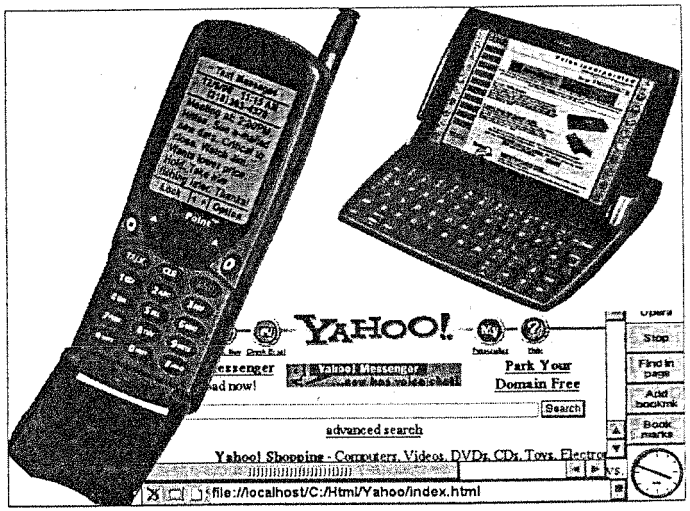


FIGURA 23.4
Dispozitive de telecomunicații fără fir cu capacități Web de la NeoPoint și Psion.

Utilizatorii cu handicap de vedere folosesc de obicei terminale cu afișaje foarte mici, uneori de o singură linie, iar adâncimea de culoare a unui terminal Braille este zero. De altfel Braille este un limbaj unic, care are reguli proprii de așezare și ortografie deci, în afară de cazul în care se poate asigura o traducere directă, va trebui să furnizați un fișier separat.

Toate acestea duc la posibilități din ce în ce mai largi pentru utilizarea SMIL la adaptarea conținutului multimedia în funcție de necesitățile și aptitudinile utilizatorului și capacitățile dispozitivului său de redare.

Capacităţile dispozitivului de ieşire

Dispozitivele de ieșire sa înmulțesc rapid, de la dispozitive de rețea la telefoane mobile cu afișaje integrate și conexiuni la rețea. Toate acestea pot să aibă capacități de afișare diferite.

Câteva dintre atributele de sistem se adresează direct capacităților dispozitivelor de ieșire:

- **`system-bitrate`** – Evaluează rata de transfer declarată de un agent utilizator față de rata de transfer cerută de un anumit element. Aceasta vă permite să încărcați fișiere mai voluminoase în cazul unei conexiuni de mare viteză și fișiere mai firave în cazul unor linii telefonice cu viteză redusă.
- **`system-required`** – Specifică numele unei extensii față de funcționalitatea de bază a SMIL. Dacă agentul utilizator suportă această extensie, elementul care o cere este ales. Este posibil ca aici să apară de fapt o zonă de nume XML.
- **`system-screen-size`** – În funcție de dimensiunea ecranului raportată de agentul utilizator, se alege o dimensiune potrivită pentru regiune și pentru fișierele multimedia.
- **`system-screen-depth`** – În funcție de adâncimea de culoare raportată a agentului utilizator, se aleg fișierele multimedia cele mai potrivite pentru afișare.

În necunoscut – modificările utilizatorilor şi tehnologiile viitorului

Cazul editorului GRiNS este edificator prin faptul că până și o specificație atât de simplă și clară ca SMIL a fost modificată de creatorii programului, astfel că multe porțiuni din SMIL-ul „lor" nu mai sunt în mod evident același lucru cu cele pe care le folosesc ceilalți. Dacă vă gândiți și la schimbările planificate de SMIL Boston, vă așteaptă o perioadă interesantă.

Din cauza conflictelor cu DOM, SMIL Boston a definit sinonime pentru `clip-begin` și `clip-end`. Deși versiunea Boston a SMIL este încă în lucru, iar detaliile se mai pot schimba, este clar că unele modificări vor rămâne până la publicare. Noile nume ale atributelor sunt `clipBegin` și `clipEnd`, o schimbare minoră, dar scopul era să se elimine cratimele. În utilizare, se comportă identic. Pentru compatibilitate, ar trebui să folosiți întotdeauna mai întâi noile nume pentru atribute, urmate de cele vechi. SMIL Boston va vedea mai întâi primele apariții și, deoarece a obținut deja informațiile necesare, va ignora al doilea set de argumente. SMIL 1.0 va ignora complet noile atribute, deoarece nu știe nimic despre acestea și le va procesa pe cele vechi.

Și celelalte modificări din Boston sunt asemănătoare, acestea referindu-se la îmbunătățirea accesibilității, a animațiilor și a capacităților multimedia, iar altele îmbunătățesc interfața. Printre îmbunătățiri este și modularizarea diferitelor componente ale specificației astfel încât porțiunile vor fi folosite în situațiile corespunzătoare, fără a consuma resurse și cu porțiunile din specificație care nu sunt necesare.

Iată câteva exemple de aplicații pentru viitor:

- Comercianții cu amănuntul care doresc să creeze un catalog sofisticat vor putea să folosească o variantă viitoare a SMIL prin care vor putea să creeze afișaje animate care se modifică în funcție de mediu și de interesele indicate de utilizator.
- Studenții vor putea să vizioneze o prezentare interactivă și să răspundă la întrebări legate de conținutul acesteia, prezentarea modificându-se în funcție de gradul în care studentul a înțeles conceptele prezentate.
- Utilizatorii care creează prezentări vor putea să aleagă căi exclusive prin prezentare după cum întrebările publicului vor dicta următorul segment al acesteia.

De fapt, SMIL este unul dintre standardele care se depanează cel mai ușor, deoarece majoritatea problemelor pe care le veți întâlni se datorează unei lipse de suport și nu unei implementări incorecte. Rămâne de văzut în ce măsură viitorii agenți vor asigura un suport pentru CSS2, deoarece toate implementările actuale ale CSS lasă mult de dorit, dar cel puțin pornesc de la un cod relativ curat. În Figura 23.1 ați văzut o reparație simplă pentru implementarea RealPlayer 7 Basic, care a constat în comentarea unei linii de cod deoarece programul de redare nu suporta un fișier `textstream`. De asemenea, ați văzut că nu sunt suportate decât un număr limitat de tipuri de fișiere audio, deci a trebuit să folosiți un fișier .wav în loc de unul MIDI. În general acestea sunt tipurile de probleme cu care vă veți confrunta și pe care va trebui să le corectați, deoarece SMIL, în sine, este destul de simplu.

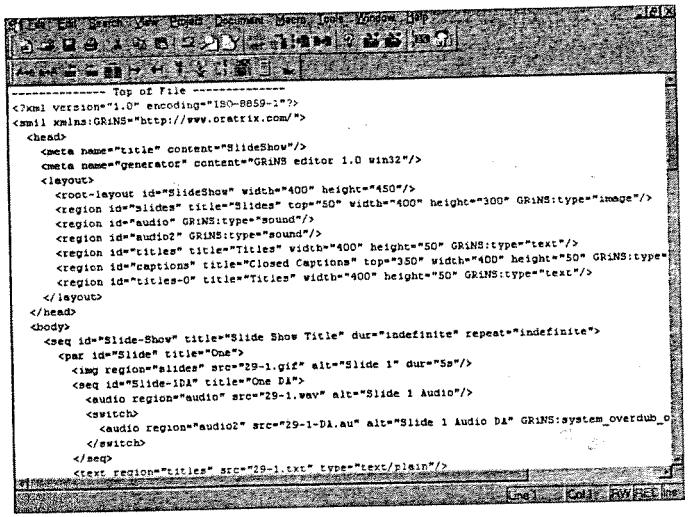
Depanarea SMIL

Prezentarea mea nu merge

După ce ați scris codul prezentării dumneavoastră, ați observat că nu se întâmplă nimic, sau ceea ce se întâmplă este departe de ceea ce vă doreați.

Pentru a rezolva problema, încercați următorii pași:

1. Vedeți vreo eroare în momentul în care încărcați codul? Căutați mai întâi în acel loc, sau în imediata sa apropiere. Dacă nu folosiți un instrument de creare care validează sau un analizor, încercați să folosiți unul. Asigurați-vă că DTD-ul standard al W3C nu a fost înlocuit cu unul patentat, deoarece nu se știe ce obiceiuri proaste veți deprinde dacă aveți un ghid înșelător.
2. Dacă nu se raportează nici o eroare, încercați să testați dacă nu cumva chiar instrumentul pe care îl folosiți conține erori. O mare parte a codului existent conține bug-uri, mai ales pentru un standard atât de „extins" cum pare SMIL. W3C lucrează deja la versiunea următoare, Boston, iar mulți producători au început deja să încorporeze porțiuni ale Boston în produsele lor, chiar înainte ca acesta să devină o recomandare oficială. Acesta poate fi de obicei depistat din DTD-ul non-standard folosit în declarația `DOCTYPE`, dar nu întotdeauna. De exemplu, GRiNS înlocuiește pe tăcute codul dumneavoastră cu ceea ce consideră el că ar trebui, inclusiv eliminând instrucțiunea dumneavoastră DTD, chiar dacă nu o folosiți pentru editare. Figura 23.5 vă prezintă ce se întâmplă dacă nu aveți grijă și lăsați GRiNS să editeze codul dumneavoastră. Observați că unele atribute sunt complet diferite de standard, atributul `xmlns` indică spre nicăieri, iar declarația `DOCTYPE`, care conține și DTD-ul, a dispărut.
3. Încercați să izolați secțiuni până când găsiți fie ceva care funcționează, fie ceva care nu funcționează. Ambele informații pot fi valoroase. Dacă găsiți o secțiune care funcționează, puteți să porniți de la aceasta și, treptat, să ajungeți la un program complet și funcțional. Dacă găsiți o secțiune care nu funcționează, puteți să încercați să o eliminați și să vedeți dacă restul se comportă ceva mai bine.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<smil xmlns:GRiNS="http://www.oratrix.com/">
  <head>
    <meta name="title" content="SlideShow"/>
    <meta name="generator" content="GRiNS editor 1.0 win32"/>
    <layout>
      <root-layout id="SlideShow" width="400" height="450"/>
      <region id="slides" title="Slides" top="50" width="400" height="300" GRiNS:type="image"/>
      <region id="audio" GRiNS:type="sound"/>
      <region id="audio2" GRiNS:type="sound"/>
      <region id="titles" title="Titles" width="400" height="50" GRiNS:type="text"/>
      <region id="captions" title="Closed Captions" top="350" width="400" height="50" GRiNS:type=
      <region id="titles-0" title="Titles" width="400" height="50" GRiNS:type="text"/>
    </layout>
  </head>
  <body>
    <seq id="Slide-Show" title="Slide Show Title" dur="indefinite" repeat="indefinite">
      <par id="Slide" title="One">
        <img region="slides" src="29-1.gif" alt="Slide 1" dur="5s"/>
        <seq id="Slide-1DA" title="One DA">
          <audio region="audio" src="29-1.wav" alt="Slide 1 Audio"/>
          <switch>
            <audio region="audio2" src="29-1-DA.au" alt="Slide 1 Audio DA" GRiNS:system_overdub_o
          </switch>
        </seq>
        <text region="titles" src="29-1.txt" type="text/plain"/>
```

FIGURA 23.5
Iată ce se întâmplă dacă lăsați editorul GRiNS să vă actualizeze codul și nu aveți o versiune de siguranță.

4. Dacă nu se vede nimic, verificați dacă fișierele sunt citite înlocuind numele unui fișier cu altceva. Analizorul ar trebui să protesteze dacă validați, dar nu se știe niciodată. Unele instrumente de creare nu validează existența unui fișier decât în momentul în care încearcă să îl citească.
5. Asigurați-vă că atributele de test, dacă există, sunt setate corect, atât în codul SMIL cât și în agentul utilizator. Recomandarea SMIL este destul de opacă în descrierea funcționării acestora, iar descrierile listate vă pot induce în eroare, deoarece nu există nici o obligație pentru numele descriptive în engleză. Atributele pentru preferințele utilizatorului trebuie să fie evaluate la o valoare identică sau într-o anumită relație între ele când sunt testate, și nu prea contează pentru ce sunt folosite, cel puțin nu în SMIL. Deși au o valoare liniștitor de explicită, nu conțin nimic care să vă asigure că vor fi folosite așa și trebuie să vă bazați doar pe buna credință a autorului. Toată logica este creată de dumneavoastră. Deci, dumneavoastră (sau altcineva) puteți folosi `system-language="fr"` pentru a porni și opri afișarea unor imagini grafice la fel de ușor ca și pentru a selecta versiunea în franceză a textului.
6. Asigurați-vă că nu încercați prea multe din prima clipă, creând un imens edificiu multimedia, cu sute de fișiere într-o coregrafie complexă de imagini în mișcare și sunete. Cele mai simple lucruri sunt într-adevăr simple și este greu de imaginat că nu ați putea să începeți cu o prezentare simplă la care să adăugați treptat elemente, în loc să încercați mai mult decât puteți duce.

CAPITOLUL

24

Utilizarea MathML

(The Math Markup Language – limbajul de marcare matematic)

În acest capitol

MathML
Utilizarea elementelor de prezentare ale MathML
Completările la marcajele de prezentare pentru MathML 2.0 propuse de W3C
Matematică prin elementele de sintaxă ale MathML
Utilizarea elementelor de conținut ale MathML
Modificările și completările la elementele de conținut propuse pentru MathML 2.0
Elementul de interfață MathML
Sintaxă sau prezentare
Combinarea matematicii și textului în MathML
Instrumente MathML
CML și alte aplicații științifice ale XML
Studii de caz

MathML

Prezentarea matematicii pe Web, prin intermediul HTML, a fost limitată la nivelul matematicii predate în gimnaziu. Pentru orice subiect mai serios trebuia să realizați și să transmiteți imagini grafice. Deși există numeroase soluții patentate, cum ar fi Microsoft Equation Editor, interoperabilitatea dintre aceste instrumente lasă mult de dorit, iar calitatea prezentării a fost sub nivelul așteptărilor comunității matematice și științifice.

MathML, limbajul de marcare matematic bazat pe XML, încearcă să vină în întâmpinarea cerințelor unei comunități diverse de utilizatori interesați de matematică, cum ar fi profesorii de matematică și de științe, cercetătorii și matematicienii activi, industria publicațiilor de matematică și știință și producătorii de instrumente pentru crearea, manipularea și afișarea ecuațiilor și construcțiilor matematice în diverse moduri. (Fără să mai amintim de utilizatorii pentru care au fost create toate acestea, cei care doresc să publice, să citească și să înțeleagă notațiile elementare din matematică, limbajul artificial universal care este la fel de bine înțeles de un chinez și de un german.)

Pentru a satisface nevoile acestor grupuri variate, a fost necesar să se creeze două limbaje de marcare bazate pe XML, în paralel, care sunt incluse ca un tot în specificaţie, un limbaj de afişare care este folosit mai ales pentru a şti cum trebuie să *arate* matematica în pagină şi un limbaj sintactic care vă indică ce *înseamnă* matematica. Acest limbaj sintactic paralel este discutat în a doua jumătate a capitolului.

Deci un utilizator de Mathematica, un mediu profesional de editare şi rezolvare de probleme matematice, poate va dori să transmită nişte fragmente matematice cuiva care foloseşte MathEQ sau MathType, acestea fiind nişte editoare matematice relativ simple. Folosind MathML, aceştia ar trebui să poată transfera informaţiile în ambele sensuri într-un mod util ambelor părţi. Deoarece matematica devine tot mai dependentă de asistenţa calculatorului, posibilitatea utilizării MathML ca format universal de transfer face posibile colaborări care înainte erau realizate după transcrieri manuale extrem de laborioase.

Standardul existent, deşi foarte util, nu acoperă nici pe departe întreaga matematică modernă. Totuşi, poate destul de uşor să-şi ducă utilizatorii prin primii câţiva ani de colegiu şi permite reprezentări şi codări matematice folosite în viaţa de zi cu zi, fără a ajunge în domeniile teoretice ale fizicii non-Newtoniene şi matematicii abstracte.

În acest capitol veţi studia în detaliu specificaţia de prezentare actuală, MathML 1.01, cu câteva comentarii legate de direcţia pe care o va lua probabil noua recomandare, MathML 2.0, şi veţi examina instrumentele folosite pentru a crea documente MathML. De asemenea, veţi afla despre un limbaj ştiinţific înrudit, CML, limbajul de marcare pentru chimie, care realizează pentru chimişti şi biologi ceea ce realizează MathML pentru matematicieni. Ca în cazul tuturor standardelor bazate pe XML, documentele şi limbajul W3C sunt referinţa cea mai autorizată şi ele trebuie consultate, mai ales pentru a controla dacă nu s-au publicat erate de la apariţia acestei cărţi – sfârşitul verii anului 2000 – şi până în momentul în care citiţi această secţiune. Documentele legate de MathML pot fi găsite la adresa `http://www.w3.org/Math/`.

Importanţa MathML

Notaţiile matematice sunt mult mai utile şi mai răspândite decât ne putem închipui. Sunt utilizate de specialişti în aproape orice domeniu imaginabil, de la sociologie la fizică nucleară. Nici una dintre ştiinţele moderne nu este accesibilă fără înţelegerea notaţiilor matematice de bază utilizate în descrierea sa; chiar şi tâmplarii au nevoie uneori de teorema lui Pitagora. Rezolvările acestor necesităţi în HTML sunt inadecvate şi grosolane, iar persoanele cu handicap de vedere sunt extrem de dezavantajate de acestea. Abilitatea de a comunica între procese şi de a indexa lucrările matematice în cercetările ştiinţifice este de asemenea stânjenită, deoarece în majoritatea cazurilor textul este suplimentat prin imagini grafice care sunt invizibile atât pentru motoarele de căutare, cât şi pentru utilizatorii care nu dispun de browsere vizuale.

Deşi MathML poate să pară încărcat faţă de eleganţa lapidară a formulelor matematice, acestea sunt menite să transmită idei complexe într-o manieră uşor de înţeles pentru oameni. Forma şi orientarea simbolurilor, fonturile utilizate şi poziţia lor faţă de alte simboluri, toate transmit informaţii care ar fi complicat de descris în cuvinte.

Deoarece MathML este destinat, în parte, pentru a fi creat şi folosit de calculatoare, are sens să „dezvăluim" într-o oarecare măsură conceptele împachetate în notaţiile matematice concise, pentru ca un calculator să le proceseze mai eficient. Aşa cum majoritatea documentelor HTML de pe Web sunt create în ziua de azi cu ajutorul unor editoare grafice care vă ajută să plasaţi corect secvenţele de etichete complexe şi ascund detaliile confuze ale HTML de utilizatorii pentru care modul de funcţionare este mai puţin important decât aspectul, la fel şi majoritatea documentelor MathML sunt create acum cu ajutorul unor editoare specializate. Nimeni nu se aşteaptă să citească efectiv codul brut MathML, la fel cum nu se aşteaptă să citească un cod HTML şi să îşi imagineze cum ar trebui să arate. Calculatoarele îl pot citi şi reda în diferite moduri, în funcţie de utilizarea sa ca intrare pentru un browser sau ca mecanism de transfer între medii matematice dedicate sau editoare de ecuaţii. Însă pentru lucruri mici, costul şi curba abruptă de învăţare a unui mediu de dezvoltare matematic dedicat, cum este Mathematica, nu sunt justificate. Dacă scrieţi un document Web despre Einstein şi doriţi să redaţi corect vestita sa formulă $E=mc^2$, scrierea manuală a unui mic fragment MathML este destul de simplă, după cum puteţi vedea în exemplul următor:

```
<math>
  <mrow>
    <mi>E</mi>
    <mo>=</mo>
    <mrow>
      <mi>m</mi><mo>&it;</mo>
      <mrow>
        <apply>
          <power />
          <ci>c</ci>
          <cn>2</cn>
        </apply>
      </mrow>
    </mrow>
  </mrow>
</math>
```

Observaţi că pentru exprimarea ridicării la putere s-au folosit câteva elemente de sintaxă (conţinut), deoarece acestea sunt redate mai natural într-un browser audio. Într-un browser vizual, nu va fi nici o diferenţă vizibilă între redarea prin elemente de prezentare şi redarea prin elemente de conţinut (sintaxă).

De asemenea, prin comparaţie cu practica răspândită de a utiliza imagini GIF sau PNG pentru a transmite notaţii matematice, care sunt complet necunoscute pentru oricine sau orice procesor care nu este capabil de recunoaştere vizuală, dimensiunea totală a fişierelor pentru paginile MathML este mult mai mică decât pentru o imagine grafică echivalentă (în ciuda relativei complexităţi a fişierului sursă). Din punct de vedere vizual, fişierul redat este (sau ar trebui să fie) exact echivalent cu o pagină tipărită sau întocmită manual.

Totuşi, fişierul MathML prezintă avantaje considerabile pentru persoanele cu handicap de vedere, prin faptul că este complet lizibil pentru browserele audio şi poate fi dimensionat dinamic sau reformatat pentru a se potrivi cu cerinţele diferiţilor utilizatori.

Deci o descriere matematică în MathML satisface cerinţe multiple, respectă un standard XML al W3C, suportă conversii între multe formate diferite, inclusiv formate patentate, dacă producătorii îl suportă ca mediu de transfer, şi este uşor de editat, căutat şi indexat pe Web. În plus, MathML se potriveşte pentru prezentarea unor fragmente matematice incluse în alte documente, în timp ce tehnologiile concurente nu pot face aşa ceva. Ca în

cazul oricărui vocabular XML încorporat, utilizarea zonei de nume MathML va permite ca MathML să fie inclus în documente XHTML sau alte documente XML, fără a include aptitudini matematice în alte standarde XML.

Sugestie de la LeeAnne.com Words to weave by

Zona de nume MathML a fost definită ca `http://www.w3.org/1998/Math/MathML`, iar această zonă de nume trebuie utilizată în orice document XML sau HTML care foloseşte structuri MathML.

Noua schiţă de lucru pentru MathML 2.0 adaugă o discuţie despre modelul document obiect (Document Object Model – DOM) al W3C şi despre modul în care acesta ar putea să facă posibilă comunicaţia dintre document şi conţinutul încorporat. Aceasta este o parte a problemei generale a comunicării între browsere şi motoarele de redare care manipulează documentele şi conţinuturile externe.

Secţiunea următoare tratează elementele de prezentare, deoarece majoritatea utilizatorilor sunt mai interesaţi de aspectul documentelor decât de înţelesul lor. Elementele sintactice sunt utile mai ales pentru mediile matematice specializate, care vă permit nu numai să editaţi şi să redactaţi formule matematice, dar şi să rezolvaţi şi să reprezentaţi grafic ecuaţii.

Utilizarea elementelor de prezentare ale MathML

Elementele de prezentare ale MathML sunt foarte asemănătoare cu limbajele TeX/LaTeX ale lui Donald E. Knuth şi cu limbajul de prezentare matematic ISO 12083, deşi sunt mult mai structurate şi mai lizibile pentru oameni. Totuşi, nici unul dintre aceste limbaje nu este deosebit de capabil să transmită informaţii de sintaxă, deci MathML reprezintă un salt considerabil în această privinţă, deoarece combină în acelaşi pachet atât elemente sintactice, care permit codarea în document a înţelesului matematic efectiv, cât şi elemente de prezentare.

Elementele de prezentare sunt suportate la nivelul cel mai larg de producătorii de editoare matematice, dar dezvoltările ulterioare vor asigura o acoperire completă a specificaţiei pentru majoritatea instrumentelor pe care le veţi folosi pentru a crea fragmente matematice pentru o eventuală prezentare pe Web.

Sugestie de la LeeAnne.com Words to weave by

Mulţi editori au investit enorm în instrumente tipografice pentru matematică, patentate sau fără standarde. Multe universităţi şi mulţi oameni de ştiinţă, de exemplu, folosesc TeX şi LaTeX, iar editorul acestei cărţi, Que, foloseşte pentru a tipări matematică, aptitudinile de machetare ale Adobe FrameMaker. Standardul MathML şi, în special, cele mai noi implementări ale sale, nu este suficient de matur pentru a investi prea mult în conversii. Totuşi, producători terţi vor crea în curând instrumente prin care conversia va fi mult mai simplă decât dacă aţi încerca să o realizaţi singur.

Toate elementele de prezentare încep cu litera „m“, poate pentru a le distinge de elementele de conţinut. În orice caz, rezultatul este că vă puteţi da seama uşor dacă un element este un element de prezentare deoarece acesta va avea la începutul numelui un „m“ *străin*. Elementele de prezentare din MathML se aseamănă cu elementele HTML prin faptul că scopul lor principal este să definească aspectul final al afişării şi nu înţelesul matematic al elementului.

De exemplu, elementul `mfenced` înconjoară din punct de vedere al algoritmului o expresie cu orice formă de acolade, paranteze drepte sau rotunde sau linii verticale, fără a transmite vreo informaţie despre sensul de bază al notaţiei, şi nici măcar dacă există vreun astfel de sens. În unele cazuri este posibil să se deducă un astfel de sens folosind ideile obişnuite despre înţelesul unei anumite notaţii, dar, deoarece MathML nu cunoaşte nimic din universul ideilor matematice în afară de cele definite prin elementele de sintaxă, nu va putea să determine în mod sigur ce înseamnă o anumită notaţie fără a folosi elemente de sintaxă.

Notă

Deşi natura străină a literei „m“ de la începutul elementelor de prezentare poate să nu fie evidentă până când nu vă familiarizaţi cu elementele MathML, diferenţa dintre `mfenced` şi cuvântul englez „fenced“ (îngrădit) este destul de clară. Are un „m“ în faţă. Deci `mfenced` este un element de prezentare. Multe din elementele MathML sunt cuvinte sau concatenări evidente de cuvinte în engleză, deci „m“ în `mfenced` este ataşat pur şi simplu. Diferenţa între `mi` şi `ci` este mai puţin clară, dar faptul că `mi` nu este un cuvânt englez obişnuit (în afara unui context muzical) vă dă un indiciu că este de fapt un nume de identificator cu o singură literă, „i“ şi cu un „m“ ataşat în faţă. Elementul `ci` cu care l-am comparat are un „c“ în faţă, care indică faptul că este vorba despre un element de sintaxă (sau conţinut), dar încă nu am ajuns la acestea.

Cu toate acestea, există obişnuinţe şi convenţii de utilizare a marcajelor de prezentare care ajută utilizatorul şi procesele să înţeleagă sensul unei anumite construcţii, deci chiar şi marcajele de prezentare, dacă sunt utilizate corespunzător, pot să vă ajute să transmiteţi înţelesuri.

De exemplu, elementul `mrow` poate fi folosit pentru a indica faptul că un număr oarecare de expresii trebuie să fie aşezate pe o singură linie. Însă acelaşi element poate fi folosit şi pentru a indica grupări logice, asemenea unor paranteze invizibile, astfel încât subexpresiile care trebuie să fie strâns legate într-o singură expresie să rămână împreună. Această abilitate de a imbrica expresiile delimitate prin `mrow` face posibilă despărţirea unei linii lungi, dacă lăţimea afişajului nu este destul de mare, punctele de separare putând fi alese preferenţial. Analog, un browser audio poate folosi delimitatori `mrow` pentru a grupa expresii într-un mod mai sugestiv decât o simplă listă.

Sugestie de la LeeAnne.com Words to weave by

Elementul `mrow` este atât de important încât multe alte elemente atribuie un element `mrow` ca şi container pentru o expresie inclusă, dacă acesta nu există.

Atributele comune pentru toate elementele MathML sunt listate în Tabelul 24.1.

Tabelul 24.1 Atribute comune în MathML

Atribute	Valori posibile
`class`	`classname`
`style`	`stylevalue`
`id`	`uniqueid`
`other`	`anycontent`

Primele trei atribute sunt furnizate pentru compatibilitate cu CSS sau alte mecanisme de stil, iar ultimul, `other`, este folosit pentru a transmite valori dependente de aplicație altor procese, fără a încălca DTD-ul MathML.

Sugestie de la LeeAnne.com Words to weave by

Deși Recomandarea MathML permite fie ghilimele, fie apostrofuri, pentru a încadra valorile atributelor, din cauza faptului că numeroase motoare de redare folosesc Java sau alte limbaje script, ghilimelele pot să dea rezultate discordante în funcție de platformă. Utilizați întotdeauna apostrofuri pentru a încadra valorile atributelor.

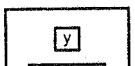
MathML afișează expresiile matematice în cutiuțe imbricate, așa cum puteți vedea în exemplul simplu din Figura 24.1. Elementul `mfrac` folosește o cutie care conține două cutii mai mici separate de o linie orizontală, în înfățișarea prestabilită. Veți vedea mai târziu că același element `mfrac` poate fi folosit pentru alte construcții matematice. Întreaga expresie `mfrac` este folosită în membrul drept al unei egalități, care are la rândul ei propria cutie împrejmuitoare și cutii pentru operatorul de egalitate și pentru membrul stâng.

Codul următor generează această expresie:

```
<math>
  <mrow>
    <mrow>
      <mi>y</mi>
      <mo>=</mo>
      <mfrac>
        <mrow>
          <mi>y</mi>
        </mrow>
        <mrow>
          <mi>x</mi>
          <mo>+</mo>
          <mn>1</mn>
        </mrow>
      </mfrac>
    </mrow>
  </mrow>
</math>
```

FIGURA 24.1

O expresie MathML și cutiile invizibile care conțin diferitele subexpresii care alcătuiesc întregul.



Elemente simbolice

Un element simbolic servește la descrierea unui simbol individual din fluxul de text al MathML. Simbolurile pot reprezenta identificatori (variabile sau constante), numere, operatori, text încorporat, spații încorporate și șiruri literale.

Tabelul 24.2 vă prezintă atributele comune pentru elementele simbolice.

Tabelul 24.2 Atributele elementelor simbolice

Atribute	Valori posibile	Valoare prestabilită
`fontsize`	`number v-unit`	`inherited`
`fontweight`	`normal | bold`	`inherited`
`fontstyle`	`normal | italic`	`normal` (cu excepția `<mi>`)
`fontfamily`	`string | css-fontfamily`	`inherited`
`color`	`#rgb | #rrggbb | html-color-name`	`inherited`

Notă

Atributele din Tabelul 24.2 sunt asemănătoare, dar nu exact echivalente în toate situațiile, cu proprietățile CSS. De exemplu, atributul `fontstyle` nu este moștenit în MathML deși proprietatea CSS cu același nume este moștenită. Analog, multe elemente MathML modifică automat valoarea atributului fontsize, deși în CSS nu există astfel de reglări automate ale proprietăților.

Elementele simbolice sunt baza prezentării, deci fiecare simbol trebuie identificat cu grijă. Singura diferență dintre funcția sinus din `sin` y și o înmulțire *sin* (*s* ori *i* ori *n*) este dată de identificarea simbolurilor individuale implicate în operație. Deși, din faptul că ultimul *sin* este scris cursiv, un om poate ușor să își dea seama că aceste litere sunt probabil una sau mai multe variabile și să ghicească din contextul expresiilor înconjurătoare sensul exact, calculatoarele nu sunt atât de inteligente. Trebuie să le ajutați mai mult. Deci următoarea listă este esențială pentru orice prezentare care va fi redată în diferite medii:

- **`<mi>` identificator** – Elementul identificator `mi` are două comportamente prestabilite. Dacă identificatorul conținut este o singură literă, aceasta este redată în mod prestabilit printr-un font cursiv. Dacă identificatorul conține mai multe litere, atunci este redat în mod prestabilit printr-un font drept. Aceste comportamente reflectă alcătuirea tipică a expresiilor matematice, în care variabilele și constantele simbolice sunt litere simple iar numele de funcții sunt alcătuite din mai multe litere. Totuși, puteți include și text arbitrar într-un element `mi`, de exemplu, punctele de suspensie vor indica un termen lipsă.
- **`<mn>` număr** – Elementul literal numeric `mn` redă conținutul ca un literal numeric. Există mai multe situații în care un element `mn` nu este potrivit, de exemplu, în jurul unui literal numeric simbolic, cum ar fi *pi* sau *e*. De asemenea, numerele complexe, imaginare, fracționare sau exponenții cer alte elemente pentru a reda operațiile matematice implicate în reprezentarea lor.
- **`<mo>` operator, paranteză sau separator** – Elementul operator `mo` înconjoară operatorii matematici, parantezele și separatorii atunci când aceștia trebuie să fie reprezentați ca operatori. Deoarece operatorii matematici au înfățișări posibile destul de complicate, inclusiv înfățișări invizibile, elementul `mo` are un număr mare de atribute pe lângă atributele prestabilite listate aici. De asemenea, utilizează un „dicționar de operatori" care este folosit de browser sau de motorul de redare. Dar mai întâi să vedem lista de atribute din Tabelul 24.3. Dacă nu se definește un comportament pentru un anumit operator, atunci se folosește valoarea prestabilită.

Tabelul 24.3 Atributele elementului <mo>

Nume	Valori	Valoare prestabilită
form	prefix | infix | postfix	stabilit prin poziția operatorului într-un <mrow> (regulă după tabel); se folosește împreună cu conținutul <mo> pentru a indexa dicționarul de operatori
fence	true | false	stabilit de dicționar (false)
separator	true | false	stabilit de dicționar (false)
lspace	number h-unit	stabilit de dicționar (.27777em)
rspace	number h-unit	stabilit de dicționar (.27777em)
stretchy	true | false	stabilit de dicționar (false)
symmetric	true | false	stabilit de dicționar (true)
maxsize	number [v-unit | h-unit] | infinity	stabilit de dicționar (infinity)
minsize	number [v-unit | h-unit]	stabilit de dicționar (1)
largeop	true | false	stabilit de dicționar (false)
movablelimits	true | false	stabilit de dicționar (false)
accent	true | false	stabilit de dicționar (false)

Dicționarul de operatori depinde de implementare și poate fi inaccesibil pentru utilizator. Regulile care se referă la operatori sunt atât de complexe încât pentru explicații detaliate ar trebui să consultați recomandarea MathML.

Pentru expresia simplă $f(x)$, pe care o pronunțați „ef de ics", echivalentul MathML poate fi <mrow><mi> f </mi><mo> ⁡ </mo><mrow><mo> (</mo><mi> x </mi><mo>) </mo></mrow></mrow>. Observați că ⁡ nu este redat vizual, dar trebuie folosit pentru a asigura redarea corectă într-un browser audio, deoarece expresia ar putea la fel de bine să se refere la „ ‚f' ori ‚x' ".

- **<mtext> text** – Elementul mtext poate fi folosit pentru a reda conținutul său ca text, dar există multe situații în care lucrul acesta nu este potrivit. De exemplu, dacă propoziția următoare ar fi încorporată în HTML, „Teorema 1: Există x astfel încât $f(x) = 0$", ar fi mai bine ca doar primul „x" și expresia „$f(x) = 0$" să fie codate în MathML pentru a vă asigura că fontul și dimensiunea acestuia se vor potrivi cu restul textului de pe pagină.

Notă Până în momentul în care se va crea un mecanism clar prin care documentele HTML și XML să comunice acest tip de informații de mediu expresiilor MathML induse, utilizarea elementului mtext va crea mai mult ca sigur, în majoritatea browserelor, insule de text care arată ciudat în mijlocul documentului. Acest mecanism reprezintă una din problemele care trebuie rezolvate în următoarea revizuire a MathML, versiunea 2.0, dar implementarea completă va necesita un acces deplin la DOM, care este la rândul său într-un proces de revizuire. Vă puteți aștepta ca toate acestea să dureze destul de mult timp.

Atenție Elementul mtext poate fi folosit și pentru a „regla" aspectul expresiilor redate de MathML, folosind entități ca   sau , dar aceasta este aproape întotdeauna o idee proastă, deoarece spațiile care sunt redate într-un anumit fel într-un browser, vor fi redate probabil diferit în altul.

- **<mspace /> spațiu** – Elementul mspace a fost creat pentru a insera un anumit tip de spațiu într-o expresie MathML. Primește atributele width, height și depth, care definesc exact cât de mare trebuie să fie spațiul. Deoarece este folosit mai ales pentru a regla aspectul expresiilor redate în MathML, utilizarea sa este aproape întotdeauna o idee proastă. Spațiul generat în acest mod probabil nu se va comporta corect dacă dimensiunea de bază a fontului este modificată pentru a se potrivi cu cerințele utilizatorului, și nici nu este sigur că este stabil pe toate platformele. O opțiune mai bună este inserarea spațiilor cu funcții invizibile, folosind elementul <mphantom>.
- **<ms> șir literal** – Elementul ms reprezintă un șir literal ce respectă convențiile obișnuite din XML pentru secvențele escape pentru caractere speciale și pentru folosirea entităților cu nume. Deoarece conținutul unui element ms este redat de obicei între apostrofuri, acestea ar trebui întotdeauna incluse prin secvențe escape. Pentru că în MathML spațiile în plus sunt unite în mod prestabilit, șirurile lungi pot fi despărțite pe mai multe linii, fără a fi necesar un tratament special. Semnele de citat folosite pot fi selectate utilizând atributele rquote și lquote, care suprascriu valorile prestabilite. Deși acest lucru este destul de comod, în alte limbi decât engleza americană ar fi mai bine să alegeți valoarea în funcție de limba documentului, așa cum *se presupune* că se întâmplă dacă folosiți eticheta <q> în HTML.

Secțiunea următoare se referă la elementele care în specificația MathML se numesc „general layout schemata" (scheme de machetare generale). *Schemata* este pluralul în engleză pentru schemă, și ca să fim sinceri, este un cuvânt de jargon destul de confuz. De fapt, înseamnă elemente de machetare.

Scheme de machetare generale

Acestea sunt elemente de prezentare mai complexe, folosite în MathML pentru a grupa diferite simboluri în diferite moduri. Înainte de a vedea ce sunt acestea, ar trebui totuși să înțelegeți ce sunt *operatorii invizibili*, care formalizează operațiile ce în mod normal se omit în reprezentarea expresiilor matematice. MathML asigură acești operatori speciali, care de obicei nu sunt redați în browserele vizuale, deoarece pot fi necesari pentru a da înțeles unei expresii redate într-un browser audio.

Un exemplu ar putea fi înmulțirea implicită din expresia $2x$. Deși se pronunță de obicei „doi ics", expresia *înseamnă* de fapt „doi ori ics". Înmulțirea este o operație atât de comună, încât majoritatea notațiilor matematice o consideră prestabilită, pentru a economisi spațiu și pentru a ușura înțelegerea expresiilor mai complexe.

Un alt exemplu este operatorul de funcție implicat în expresia $f(x)$. Utilizarea acestei notații este destul de ambiguă, deoarece se poate confunda funcția f cu o variabilă f, dacă nu sunteți atent la cursivitatea literelor.

Din nou, uneori numerele și variabilele sunt juxtapuse pentru concizie în unele notații, deși sunt disjuncte din punct de vedere logic. MathML dispune de câțiva operatori speciali invizibili, care ar trebui să fie utilizați pentru a clarifica relația exactă dintre aceste elemente, așa cum se vede în Tabelul 24.4.

Tabelul 24.4 Operatorii MathML invizibili

Nume complet	Nume prescurtat	Situația în care este folosit
⁢	⁢	$2x$
⁡	⁡	$f(x)$ sau $\sin y$
⁣	⁣	m_{12}

Lista este scurtă, dar de fapt nu există chiar atât de multe situații în care o operație este subînțeleasă și nu este scrisă.

- **<mrow>** – Grupează orizontal un număr oarecare de subexpresii care trebuie tratate ca o singură expresie.
  Un element mrow este folosit pentru a grupa un număr oarecare de subexpresii și, mai mult, pentru a da indicii browserelor și motoarelor de redare despre structura efectivă a ecuației modelate. Dacă o expresie mai lungă trebuie despărțită pe mai multe linii, sau dacă un browser audio trebuie să afle modul în care să grupeze descrierile, elementele mrow imbricate corect sunt cruciale. Următorul exemplu:

  ```
  <mrow><mn>2</mn><mo>&it;</mo><mi>x</mi></mrow>
  ```

  afișează expresia $2x$. Observați folosirea unui operator invizibil.
- **<mfrac>** – Formează o fracție din două expresii, numărător și numitor.
  Elementul mfrac primește un atribut linethickness, care permite elementului să fie folosit la notații fracționare, de exemplu pentru expresii binomiale (care au linethickness setat la valoarea zero pentru ca linia de fracție să nu fie afișată) sau pentru fracții imbricate (care pot folosi o linie mai groasă pentru fracția exterioară). Exemplul:

  ```
  <mfrac><mn>1</mn><mn>2</mn></mfrac>
  ```

  afișează fracția 1/2.
- **<msqrt>** – Formează o rădăcină pătrată dintr-una sau mai multe expresii. Exemplul următor:

  ```
  <msqrt><mi>x</mi></msqrt>
  ```

  afișează $\sqrt{x}$.
  Dacă există mai multe expresii, se presupune că există un element mrow implicit, care le conține.
- **<mroot>** – Formează un radical din exact două expresii, bază și indice.
  Este o eroare dacă se includ mai mult sau mai puțin de două expresii. Exemplul:

  ```
  <mroot><mn>3</mn><mi>x</mi></mroot>
  ```

  afișează $\sqrt[3]{x}$.
- **<mstyle>** – Modifică stilul unui număr de subexpresii. Exemplul:

  ```
  <mstyle color="blue"><mi>x</mi></mstyle>
  ```

  face ca variabila x să fie afișată în albastru, deși acest lucru ar fi invizibil pentru un browser audio sau pe un dispozitiv cu ecran monocrom și dificil de deosebit de alte culori pentru persoanele cu handicap în distingerea culorilor.
  Elementul mstyle poate primi orice atribut acceptat de un element de prezentare MathML. Dacă există mai multe argumente, acestea sunt tratate ca și când ar fi închise într-un element mrow implicit. Pe lângă atributele acceptate de orice element de prezentare, mstyle acceptă și transmite mai departe următoarele atribute, moștenite implicit de orice element MathML ca parte a mediului în care operează. Acestea sunt prezentate în Tabelul 24.5.

Tabelul 24.5 Atributele elementului <mstyle>

Nume	Valori	Valoare prestabilită
scriptlevel	['+' | '-']	întreg fără semn moștenit
displaystyle	true | false	moștenit
scriptsizemultiplier	number	0.71
scriptminsize	number v-unit	8pt
color	#rgb | #rrggbb | html-color-name	moștenit
background	#rgb | #rrggbb | transparent | html-color-name	transparent

MathML folosește două atribute, displaystyle și scriptlevel pentru a controla unele caracteristici ale prezentării. Efectul principal al atributului displaystyle este că influențează efectele celorlalte atribute, cum ar fi atributele largeop și movablescripts ale <mo>. Efectul principal al atributului scriptlevel este controlul dimensiunii fontului. De obicei, cu cât scriptlevel este mai mare, cu atât fontul este mai mic. (Motoarele de redare non-vizuale pot răspunde la dimensiunea fontului într-un mod corespunzător mediului.) Agenții de redare mai sofisticați pot alege să folosească aceste atribute în alte moduri, cum ar fi redarea expresiilor cu displaystyle='false' într-o manieră dreaptă și compactă.

- **<merror>** – Include un număr oarecare de mesaje de eroare de sintaxă care provin de la preprocesor. Se presupune că aceste etichete vor fi inserate chiar de preprocesor; este greu să ne imaginăm un motiv pentru care cineva ar dori să le insereze din proprie inițiativă într-un document.
  Dacă sunt incluse mai multe argumente, conținutul este tratat ca și când ar fi inclus într-un element mrow implicit.
- **<mpadded>** – Ajustează spațiul din jurul argumentelor pe care le conține, folosind atributele prezentate în Tabelul 24.6. Acest element, ca și elementul mspace, trebuie folosit cu foarte multă grijă (dacă chiar doriți să îl folosiți) deoarece este aproape sigur că va da rezultate diferite în browsere diferite, iar dacă folosiți mecanisme de stil, va afecta în diverse moduri afișarea expresiilor.
  Acest element nu are efect asupra dimensiunii de afișare a expresiei incluse ci doar asupra marginilor din jurul acesteia.
  Elementul <mpadded> acceptă oricâte argumente; dacă numărul acestora nu este 1, conținutul este tratat ca și când ar fi inclus într-un element mrow implicit.

Tabelul 24.6 Atributele elementului <mpadded>

Nume	Valori	Valoare prestabilită
width	[+ | -] unsigned-number (% [pseudo-unit] | pseudo-unit | h-unit)	la fel cu conținutul
lspace	[+ | -] unsigned-number (% [pseudo-unit] | pseudo-unit | h-unit)	0
height	[+ | -] unsigned-number (% [pseudo-unit] | pseudo-unit | v-unit)	la fel cu conținutul
depth	[+ | -] unsigned-number (% [pseudo-unit] | pseudo-unit | v-unit)	la fel cu conținutul

Atributele `width` și `lspace` nu sunt simetrice. `lspace` se referă la spațiul de la început (marginea din stânga), în timp ce atributul `width` se referă la lățimea totală a cutiei care include conținutul. Analog, `height` și `depth` nu sunt simetrice. `height` este înălțimea totală a cutiei care include conținutul, în timp ce `depth` este spațiul vertical dintre cutie și linia de bază.

Atenție Ca în cazul tuturor elementelor care inserează spații, nu este recomandat să folosiți elementul `mpadded` pentru a regla aspectul pentru un anumit browser sau motor de redare.

- **<mphantom>** – Face conținutul invizibil, dar îi păstrează dimensiunea.
  Este folosit pentru a alinia elementele dacă unul dintre ele lipsește din expresia efectivă. Prin crearea unui element fantomă, elementele din jur se vor așeza cu spații corecte, fără să afecteze în alt mod redarea expresiei.
- **<mfenced>** – Înconjoară conținutul alcătuit dintr-un număr oarecare de expresii cu o pereche de paranteze. Exemplul

```
<mfenced><mi>x</mi></mfenced>
```

  afișează (x).
  Elementul `mfenced` își înconjoară argumentul (argumentele) cu un anumit tip de paranteze, acolade, paranteze rotunde, bare verticale sau altceva. Dacă sunt mai multe, argumentele sunt separate de valoarea atributului `separators`, dacă acesta există. Dacă există mai mulți separatori, lista se aplică secvențial. Ultimul separator este repetat, dacă este necesar (vezi Tabelul 24.7).

Tabelul 24.7 Atributele elementului `mfenced`

Atribute	Valori	Valoare prestabilită
open	string	(
close	string	)
separators	character *	,

Acestea sunt elementele cel mai des folosite în MathML; totuși, în secțiunea care urmează vom discuta elementele care cer o afișare ceva mai complexă, indici sau limite, care apar de obicei ca numere micuțe scrise deasupra sau dedesubtul expresiei. Din nou, cuvântul de jargon *schemata* complică puțin lucrurile, dar din descrierea lor se vede că sunt pur și simplu elemente, ca oricare alte elemente MathML.

Scheme pentru indici și limite

- **<msub>** – Atașează un indice inferior la o bază; bază și indice inferior. Exemplul

```
<msub><mi>x</mi><mi>y</mi></msub>
```

  duce la afișarea x_y.
  Elementul `msub` primește un atribut `subscriptshift`, care permite modificarea deplasării exacte a indicelui față de valoarea prestabilită, care este automată și de obicei are valoarea înălțimii literei normale.
- **<msup>** – Atașează un indice superior la o bază; bază și indice superior. Exemplul

```
<msup><mi>x</mi><mi>y</mi></msup>
```

  duce la afișarea x^y.
  Elementul `msup` primește un atribut `superscriptshift`, care permite modificarea deplasării exacte a indicelui față de valoarea prestabilită, care este automată și de obicei are valoarea înălțimii literei normale. Deoarece indicii superiori și inferiori se pot aplica și la expresii întregi, este important să structurați corect codul folosind elemente `mrow`, pentru a vă asigura că se comportă corect în toate motoarele de redare și în toate browserele. Deci pentru a afișa $(x+y)^2$, următorul fragment de cod se va comporta corect:

```
<msup>
  <mrow>
    <mo> ( </mo>
    <mrow>
      <mi> x </mi>
      <mo> + </mo>
      <mi> y </mi>
    </mrow>
    <mo> ) </mo>
  </mrow>
  <mn> 2 </mn>
</msup>
```

- **<msubsup>** – Atașează o pereche indice inferior, indice superior la o bază; bază, indice inferior și indice superior. Exemplul

```
<msubsup><mi>x</mi><mi>y</mi><mi>z</mi></msubsup>
```

afișează x cu indicele inferior $_y$ și cu indicele superior ^{z} poziționate corect unul deasupra celuilalt.

Elementul msubsup primește atributele subscriptshift și superscriptshift, care permit modificarea deplasării exacte a indicilor în ambele direcții față de valoarea prestabilită, care este automată și de obicei are valoarea înălțimii literei normale. Deși poate să pară redundant, redarea acestui element aliniază indicii corect unul sub altul și asigură o distanțare egală față de elementul bază. Dacă utilizați consecutiv msup și msub, indicii vor fi deplasați unul față de celălalt.

- **<munder>** – Atașează un subindice unei baze; bază și subindice. Subindicii și supraindicii se comportă similar cu indicii inferiori și superiori, dar sunt folosiți pentru a indica lucruri ca limitele unei sume. Pentru a menaja editorul de complicațiile tipăririi acestor lucruri, vom pretinde că știm cum arată dacă chiar avem nevoie de ele. Deși este asemănător cu elementul msub, subindicele este plasat direct dedesubtul bazei. Subindicele primește un atribut accentunder cu valorile true sau false, care indică apropierea la care va fi redat subindicele față de bază.
- **<mover>** – Atașează un supraindice unei baze; bază și supraindice.
  mover este asemănător cu elementul msup, dar supraindicele este plasat direct deasupra bazei. Primește un atribut accentover cu valorile true sau false, care indică apropierea la care va fi redat supraindicele față de bază.
- **<munderover>** – Atașează o pereche subindice, supraindice unei baze; bază, subindice și supraindice.
  Se aseamănă cu elementul msubsup, dar subindicele este direct dedesubtul bazei iar supraindicele direct deasupra. Primește atributele accentunder și accentover cu valorile true sau false, care indică apropierea la care vor fi redați subindicele și supraindicele față de bază.
- **<mmultiscripts>** – Atașează unul sau mai mulți indici anteriori și indici de tensor unei baze; (base (subscript superscript) * [<mprescripts /> (presubscript presuperscript) *]). Acesta este un element complex care permite ca un număr oarecare de indici inferiori, superiori și anteriori să fie atașați unui argument. Elementul vid mprescripts este un marcator opțional al indicilor anteriori, deoarece aceștia sunt destul de rari, și trebuie să fie inclus în elementul mmultiscripts. Dacă nu este prezent, atunci nu există indici anteriori. Indicii care nu sunt scriși trebuie să fie identificați cu elementul vid <none />. Acest element complex este folosit la notațiile de tensori și pentru alte câteva notații. Dacă înțelegeți ce sunt tensorii, atunci știți cum arată.

Tabele și matrice

Tabelele MathML sunt mai mult influențate de SGML decât de XML, deoarece au numeroase variante prescurtate care vă permit să omiteți rândurile și elementele câmp de date care vor fi deduse. Posibilitatea ca elementele să fie deduse este contrară practicii obișnuite în XML, dar se întâlnește adesea în SGML. Această abordare face dificilă realizarea unor motoare de redare robuste pentru MathML, dar reflectă convențiile comune în notațiile matematice, care omit informații importante ce pot fi deduse din context de un cititor avizat. De exemplu, un element mtr care lipsește implică existența unui singur rând, în timp ce lipsa unui element mtd implică un element mrow care conține toate elementele din interior, dacă sunt mai multe.

- **<mtable>** – Tabel sau matrice alcătuită din zero sau mai multe rânduri.
  Acest element se comportă cam ca un tabel HTML și primește atribute similare cu cele ale tabelelor HTML. Este folosit pentru a reda orice tablou sau matrice cu aspect de tabel (vezi Tabelul 24.8).

Tabelul 24.8 Atributele elementului <mtable>

Nume	Valori	Valoare prestabilită
align	(top | bottom | center | baseline | axis) [rownumber]	axis
rowalign	(top | bottom | center | baseline | axis) +	baseline
columnalign	(left | center | right) +	center
groupalign	group-alignment-list-list	{left}
alignmentscope	(true | false) +	true
rowspacing	(number v-unit) +	1.0ex
columnspacing	(number h-unit) +	0.8em
rowlines	(none | solid | dashed) +	none
columnlines	(none | solid | dashed) +	none
frame	none | solid | dashed	none
framespacing	number h-unit number v-unit	0.4em 0.5ex
equalrows	true | false	true
equalcolumns	true | false	true
displaystyle	true | false	false

Valoarea group-alignment-list-list conține o listă de liste de valori de aliniere care trebuie să fie aplicată în ordine la toate rândurile și la toate elementele de date.

- **<mtr>** – Afișează un singur rând dintr-un tabel sau dintr-o matrice alcătuită din 0 sau mai multe elemente de tabel (vezi Tabelul 24.9).
  Este, în mare, echivalent cu elementul HTML <tr>. Dacă lipsește, atunci implicit există un singur rând.

Tabelul 24.9 Atributele elementului <mtr>

Nume	Valori	Valoare prestabilită
rowalign	top | bottom | center | baseline | axis	moștenită
columnalign	(left | center | right) +	moștenită
groupalign	group-alignment-list-list	moștenită

- **<mtd>** – Afișează o singură intrare dintr-un tabel sau dintr-o matrice (vezi Tabelul 24.10).
  Este, în mare, echivalent cu elementul HTML <td>. Dacă lipsește, atunci implicit există o singură intrare. Dacă există mai multe expresii, un element implicit mrow le include

pe toate. Valoarea group-alignment-list-list conține o listă de liste de valori de aliniere care trebuie să fie aplicată în ordine la toate elementele de date.

Tabelul 24.10 Atributele elementului <mtd>

Nume	Valori	Valoare prestabilită
rowspan	number	1
columnspan	number	1
rowalign	top | bottom | center | baseline | axis	moștenită
columnalign	left | center | right	moștenită
groupalign	group-alignment-list	moștenită

- **<maligngroup />** – Specifică alinierea.
  Atributul groupalign al maligngroup poate să ia una dintre aceste patru valori: left, center, right și decimalpoint. Valoarea prestabilită este moștenită.
- **<malignmark />** – Specifică un marcaj de aliniere alternativ.
  Dacă este specificată o aliniere decimalpoint dar caracterul după care se face alinierea ar trebui să fie orice altceva decât punctul zecimal, elementul malignmark funcționează ca un înlocuitor al punctului zecimal. Atributul edge poate fi folosit pentru a selecta caracterul care trebuie aliniat. Dacă edge='right', alinierea se face la marginea din dreapta a caracterului aflat la stânga marcajului. Dacă edge='left', alinierea se face la marginea din stânga a caracterului aflat la dreapta marcajului.

Atributul edge al malignmark poate lua două valori, right sau left. Valoarea prestabilită este left.

Expresii de activare

Poate influențat de Dynamic HTML, MathML vă permite să asociați explicit acțiuni unei subexpresii folosind un singur element maction. Desigur,se pot realiza mai multe cu CSS și XSL, dar acest element garantează niște capacități minime pentru orice implementare, chiar dacă nu suportă nici un fel de foaie de stil.

Elementul <maction> leagă una sau mai multe acțiuni de o subexpresie folosind atributul actiontype.

De exemplu, ați putea include o casetă de ajutor folosind actiontype='tooltip' sau perechea de atribute actiontype='highlight' other='#ff0000'. Atributul other este definit pentru aproape orice element ca o conexiune pentru extinderea capacităților MathML. Rețineți că nici un browser sau motor de redare nu este obligat să suporte un anumit tip de acțiune și poate să ignore de asemenea și atributul other. Valorile celor două atribute nu sunt definite în specificația MathML, iar suportul pentru o anumită valoare este în întregime dependent de implementare.

Completările la marcajele de prezentare pentru MathML 2.0 propuse de W3C

MathML 2.0 este în stadiul de finalizare, deși poate că recomandarea va fi publicată în momentul în care citiți această carte. Oricare sau toate modificările listate în acest rezumat pot fi eliminate, completate sau complet modificate înainte ca standardul să fie finalizat.

Deși specificația MathML 2.0 se află într-un stadiu preliminar, până acum sunt surprinzător de puține modificări propuse la specificația MathML 1.0, printre care se numără și următoarele elemente noi:

- **<menclose>** – Un element de îngrădire care redă un tip de notație care a fost în mod surprinzător omis de recomandarea originală MathML. Valoarea longdiv este semnul de împărțire învățat în școala generală, iar valoarea actuarial este imaginea sa în oglindă, semn folosit în domeniul asigurărilor. Valoarea radical redă echivalentul elementului msqrt.
  Elementul menclose primește atributul notation, care poate avea valorile longdiv, actuarial sau radical. Valoarea prestabilită este longdiv.
- **<meqno>** – Introduce un spațiu pentru un număr de ecuație care poate fi utilizat pentru localizare sau pentru a transforma automat locațiile în numere.
  Locația poate să înconjoare un număr de ecuație explicit. Deoarece XML permite ca porțiuni de document să fie incluse în alte documente, este posibil ca un număr explicit corect să fie incorect când este redat în context. Schema efectivă de numerotare este în afara domeniului MathML și trebuie să fie realizată prin intermediul XSLT sau al altui limbaj de transformare.
  Elementul meqno primește atributul eqno, care poate să aibă valorile string sau empty. Valoarea prestabilită este string.
  Atributul eqno permite ca referințele la ecuațiile numerotate să fie procesate automat precizând un șir oarecare pentru a identifica numărul locației. XSLT poate fi folosit pentru a transforma documentele care includ ecuații numerotate într-o serie de ținte pentru legături, de exemplu, pentru a căuta un șir astfel încât elementul inclus să poată fi înlocuit.
- **<mglyph />** – Elementul mglyph vă permite să includeți simboluri folosite în notațiile matematice de astăzi fără să vă bazați pe răspunsul, care adesea este atât de lent, al comitetelor de standardizare, cum ar fi Unicode Consortium, care au perioade de comentare si distribuție foarte lungi înainte de a permite standardizarea și utilizarea internațională a noilor simboluri. Deși acest comportament este potrivit pentru limbi, care introduc foarte rar noi caractere, matematica introduce frecvent caractere noi, pe măsură ce frontierele gândirii matematice se extind.
  Elementul mglyph primește trei atribute obligatorii:
  - alt, care trebuie să conțină o reprezentare alternativă a simbolului, care să poată fi folosită în browsere audio sau oriunde simbolul original nu este disponibil.
  - fontfamily, care este un șir oarecare, care poate fi numele unui font efectiv sau numele unei familii de fonturi în stilul CSS.
  - index, care este o valoare numerică utilizată pentru a selecta un singur caracter din fontul indicat.

Exemplul

```
<mglyph alt="[aleph]" fontfamily="HebrewLetters" index="1" />
```

poate să se refere la caracterul alef din alfabetul ebraic.

- **<mchar />** – Elementul mchar vă permite să referiți caractere non-ASCII fără să folosiți entitățile XML denumite, deoarece acestea nu sunt sigure când folosiți un browser care nu validează. Elementul mchar primește un singur atribut obligatoriu, name, care trebuie să se refere la o listă de nume descrisă chiar în specificația MathML. Exemplul

```
<mi><mchar name="aleph" /></mi>
```

se afișează sub forma caracterului ebraic alef, care reprezintă de obicei transfinitul.

Din nou, modificările exacte de sintaxă planificate pentru MathML 2.0 nu sunt încă hotărâte, dar schimbările indicate aici au fost stabile destul de mult timp și cel puțin indică direcția în care se pare că se mișcă grupul de lucru. Modificările cele mai importante de până acum au fost adăugarea unor elemente și atribute care au fost omise, corectarea unor deficiențe ale specificației originale și armonizarea specificației cu realitatea experiențelor dificile ale celor care au implementat MathML 1.0.

Matematică prin elementele de sintaxă ale MathML

În secțiunile anterioare ați studiat elementele de prezentare ale MathML, care vă permit să publicați materiale matematice într-o manieră inteligibilă. Totuși, matematica a trecut de la zilele în care totul era scris pe hârtie la o eră în care calculatoarele realizează în mod curent munca obositoare asociată cu reprezentarea grafică a funcțiilor complexe, rezolvarea problemelor matematice dificile și a altor sarcini supărătoare care, dacă ar fi realizate manual, ar încetini matematicienii actuali până la o oprire completă.

Introducerea Mathematica sau a altor medii matematice a adus o simplificare a procesului de *realizare* a matematicii superioare tot atât de mult cât au simplificat calculatoarele de buzunar aritmetica obișnuită.

În acest tip de mediu, elementele de prezentare nu sunt nici pe departe suficiente, deoarece acestea nu transmit nici un înțeles în afara celui pe care un cititor uman poate să îl deducă din cunoștințele sale matematice anterioare. Astfel, litera grecească gama (Γ) nu înseamnă nimic special decât în cazul în care cunoașteți contextul în care a fost utilizată. Calculatoarele au nevoie de un ajutor mult mai substanțial, deoarece ele de fapt nu înțeleg deloc matematica; ele aplică pur și simplu diferite reguli unor descrieri exacte pe care le pot manipula conform unor algoritmi interni.

Elementele sintactice, sau de conținut, ale MathML rezolvă această problemă prin definirea unei serii de construcții matematice pe care autorii le pot extinde aproape după plac prin definirea sau redefinirea unor construcții particularizate cu ajutorul unor comportamente externe și al unor descrieri semantice.

Utilizarea elementelor de conținut ale MathML

Elementele de conținut (sau de sintaxă) descriu sensul matematic într-un mod care poate fi manipulat, prin programare, și afișat. Deci, ca exemplu simplu, nu numai că puteți

vedea cum arată o ecuație, dar puteți afișa o reprezentare grafică a variației funcției. Potențialul pentru educație și munca în colaborare este evident, dar mai mult, ieșirea unui astfel de program poate fi afișată și pe dispozitive de ieșire alternative, cum sunt cele tactile pentru utilizatorii orbi. Astfel de reprezentări tactile se realizează în prezent manual, printr-o muncă laborioasă, și sunt limitate doar la funcțiile elementare care ilustrează conceptele matematice simple în manuale pentru persoanele cu handicap.

Elemente sintactice simbolice

Elementele sintactice simbolice sunt similare cu echivalentele lor de prezentare, cu câteva diferențe care provin din faptul că sintaxa diferă de prezentare. Deci un element mn așa cum a fost descris în secțiunile precedente poate fi văzut ca un șir ce reprezintă un număr, în timp ce un element cn *este* un număr.

- **<cn>** – O constantă numerică. Exemplul următor

```
<cn>42</cn>
```

este numărul 42.

Spre deosebire de elementul mn, care se referă doar la un conținut care ar trebui redat *ca* un număr, elementul cn se referă fără nici o ambiguitate la o constantă numerică ce poate fi folosită și manipulată ca *fiind* un număr.

Atributele elementului <cn>

- type: real | integer | rational | complex-cartesian | complex-polar | constant
- base: number (CDATA în DTD-ul XML) între 2 și 36.
- definitionURL: Un URL care indică o definiție semantică alternativă.

În mod prestabilit, o constantă numerică este un număr real cu semn, scris în baza 10, dar se poate alege o altă bază cu o valoare între 2 și 36, inclusiv. Motivul pentru care valorile bazelor sunt limitate nu sunt chiar evidente. Dacă sunt necesare mai multe părți pentru a crea un număr complex, acestea vor fi separate prin elemente vide <sep />.

- **<ci>** – Un identificator simbolic, de exemplu

```
<ci>x</ci>
```

Atributele elementului <ci>

- type: integer | rational | real | float | complex | complex-polar | complex-cartesian | constant | orice nume de element container (de exemplu un vector) sau valori de tipul său.
- definitionURL: URL

Atributul definitionURL permite ca semnificația prestabilită să fie înlocuită cu una alternativă.

Elemente de conţinut de bază

Elementele de conţinut de bază din MathML permit aplicarea unor funcţii de bază unui anumit număr de argumente. Aceste elemente tind să apară des în documentele MathML deoarece definesc relaţii simple între alte elemente:

- **<apply>** – Aplică un operator matematic argumentelor sale. Sintaxă: operator şi zero sau mai multe argumente. Exemplul următor, care reprezintă 2+2, este evaluat la 4:

```
<apply> <plus /> <cn>2</cn> <cn>2</cn> </apply>
```

Semnificaţia `apply` impune o notaţie cu operator prefixat în care, în mod inerent, nu sunt necesare paranteze.

Notă Deşi există mai mulţi algoritmi disponibili pentru evaluarea expresiilor fără paranteze, în cazul notaţiei prefixate, operatorul şi argumentele sale sunt evaluate în ordinea citirii lor din fluxul de intrare. De câte ori este întâlnit un operator în fluxul de intrare, acesta marchează începutul unei noi expresii complete. În cazul notaţiei prefixate, orice expresie sau subexpresie anterioară este în acel moment completă şi începe una nouă. Omniprezentele calculatoare ştiinţifice şi financiare Hewlett Packard (HP) folosesc tot o notaţie prefixată fără paranteze, numită notaţie poloneză inversă, care este cunoscută mai ales ca fiind bazată pe stivă, spre deosebire de citirea directă a fluxului de intrare, dar conceptele generale sunt aceleaşi.

- **<reln>** – Aplică un operator relaţional argumentelor sale. Sintaxă: operator şi argumente. Exemplul următor reprezintă expresia booleană 2=2 şi este evaluat la `true`:

```
<reln><eq /> <cn>2</cn>15 <cn>2</cn> </reln>
```

O relaţie este folosită într-un mod puţin diferit faţă de o funcţie, deci are un element de sintaxă distinct. Distincţia poate să dispară în MathML 2.0, iar utilizarea `reln` va fi poate descurajată. Este evident din studierea exemplelor că notaţiile prefixate folosite atât pentru `apply` cât şi pentru `reln` sunt identice, deci distincţia dintre acestea este într-o oarecare măsură o practică artificială, în ciuda diferenţei logice; în MathML 2.0 `<apply>` este folosit pentru toate funcţiile.

- **<fn>** – Identifică o funcţie. De exemplu, codul următor descrie funcţia simplă 2+2:

```
<fn><apply> <plus /> <cn>2</cn> <cn>2</cn> </apply></fn>
```

Acest element primeşte un atribut `definitionURL`, care permite suprascrierea sensului prestabilit cu un sens alternativ, dar nu face prea mult dacă nu este asigurat un sens alternativ, deoarece din context este evident că `<plus />` este folosit ca funcţie. MathML 2.0 descurajează folosirea acestui element în favoarea lui `csymbol`.

- **<interval>** – Defineşte un interval pe axa numerelor reale. De exemplu, codul următor identifică intervalul 5..10:

```
<interval><cn>5</cn> <cn>10</cn></interval>
```

Acest element primeşte un atribut `closure`, care poate lua una din valorile `open`, `closed`, `open-closed` şi `closed-open`. Valoarea prestabilită este `closed`. Această funcţie aşteaptă fie două elemente copil care sunt evaluate ca numere reale, fie un singur element copil care este o condiţie ce defineşte intervalul.

- **<inverse />** – Inversa generală a unei funcţii. Exemplul următor

```
<apply><inverse /><sin/> <ci>x</ci> </apply>
```

corespunde lui $\sin^{-1}$ x, arcsinus de x.

Nu toate funcţiile au o funcţie inversă unică. Acest element primeşte un atribut `definitionURL`, care permite ca sensul prestabilit să fie suprascris, dacă este necesar, cu un sens alternativ.

- **<sep />** – Separă părţile unui număr complex sau raţional. De exemplu, codul următor identifică numărul complex (2, 8*i*):

```
<cn type="complex">2<sep />8</cn>
```

Elementul `sep` este folosit în interiorul unui element `cn` pentru a permite reprezentarea părţilor componente astfel încât aplicaţia să le poată asambla.

- **<condition>** – Pune o condiţie asupra unei variabile independente sau asupra unui identificator independent. Exemplul

```
<condition><apply><in /><ci>x</ci><ci type="set">Y</ci></apply></condition>
```

defineşte o condiţie prin care variabila *x* poate lua valori numai din mulţimea *Y*.

Elementul `condition` este folosit atunci când nu puteţi enumera efectiv valorile posibile. Elementul conţine fie un singur element `reln`, fie un element `apply` folosit pentru a construi o condiţie compusă. Reţineţi că utilizarea `reln` va fi probabil descurajată în MathML 2.0.

- **<declare>** – Stabileşte numele unui identificator şi valorile unor atribute opţionale. De exemplu, iată ce trebuie să faceţi pentru a declara o mulţime *X*:

```
<declare type="set"> <ci>X</ci> </declare>
```

`declare` are următoarele atribute:

  - `type` defineşte tipul de element MathML al identificatorului declarat în acest element.
  - `scope` defineşte domeniul de vizibilitate al declaraţiei MathML.
  - `nargs` defineşte numărul argumentelor unei funcţii declarate într-un document sau într-un fragment MathML inclus.
  - `occurence` defineşte utilizarea unui operator ca ‚prefix', ‚infix' sau ‚function-model'. Aceste indicaţii permit unui motor de redare să decidă când să realizeze operaţia efectivă indicată.
  - `definitionURL` defineşte un URL care indică semnificaţia detaliată necesară a funcţiei matematice sau logice declarate.

Elementul `declare` are două roluri. Primul este să schimbe sau să specifice valorile prestabilite ale atributelor pentru anumite obiecte matematice. Al doilea este să stabilească o asociere între un nume şi un obiect folosind un element `ci`.

- **<lambda>** – Construieşte o funcţie definită de utilizator. Exemplul

```
<lambda><apply><ci>F</ci> <apply><fn><ci>f</ci></fn>
<ci>x</ci></apply></apply></lambda>
```

defineşte `[lm](F(f(x)))`.

Elementul `lambda` este folosit cu o expresie şi una sau mai multe variabile independente pentru a construi o funcţie definită de utilizator. Aceasta este o modalitate de a construi o funcţie care vi se va părea destul de utilă, fără complicaţiile creării unui simbol şi a unei notaţii noi şi fără a difuza rezultatele în lume. Notaţia `lambda` este echivalentul matematic al unei rutine sau proceduri ad-hoc dintr-un limbaj de programare.

- **<compose />** – Compune o funcție. Exemplul

```
<apply><compose /><fn><ci>f</ci></fn> <fn><ci>g</ci></fn></apply>
```

definește f _ g.
Elementul compose este operatorul de compunere a funcțiilor. Acest element primește un atribut definitionURL, care permite ca sensul prestabilit să fie suprascris, dacă este necesar, cu un sens alternativ.
- **<ident />** – Funcția identitate. Exemplul

```
<ident />
```

este redat ca id.
Pentru a suprascrie semnificația minimă prestabilită a elementului <ident /> sau pentru a-i asocia o definiție mai specială, folosiți atributul definitionURL.

Aritmetică, algebră și logică

Următoarele elemente alcătuiesc o selecție a funcțiilor obișnuite care se comportă așa cum vă așteptați. Fiecare element primește atributul definitionURL, care vă permite să suprascrieți semnificația prestabilită cu o semnificație alternativă. Elementele aritmetice sunt în mod prestabilit în baza zece:

- **<quotient />** – Câtul unei împărțiri. Este un operator binar.
- **<exp />** – Funcția exponențială. Este un operator unar.
- **<factorial />** – Funcția factorial. Este un operator unar.
- **<divide />** – Operatorul de împărțire. Este un operator binar.
- **<max />** – Maximul elementelor incluse. Este un operator *n*-ar.
- **<min />** – Minimul elementelor incluse. Este un operator *n*-ar.
- **<minus />** – Operatorul de negare sau de scădere. Este fie un operator unar de semn, fie un operator binar de scădere.
- **<plus />** – Operatorul de adunare. Este un operator *n*-ar.
- **<power />** – Operatorul de ridicare la putere. Este un operator binar.
- **<rem />** – Restul unei împărțiri. Este un operator binar.
- **<times />** – Operatorul de înmulțire. Este un operator binar.
- **<root />** – Extrage rădăcina argumentelor incluse. Primul argument, dacă există, este un element degree, care indică ordinul rădăcinii incluse. Dacă este absent, se presupune că are valoarea 2.
- **<gcd />** – Operatorul cel mai mare divizor comun. Este un operator *n*-ar.
- **<and />** – Operatorul logic „și". Este un operator *n*-ar.
- **<or />** – Operatorul logic „sau". Este un operator *n*-ar.
- **<xor />** – Operatorul logic „sau exclusiv". Este un operator *n*-ar.
- **<not />** – Operatorul logic de negare. Este un operator unar.
- **<implies />** – Operatorul relațional logic „implică". Este un operator binar.
- **<forall />** – Operatorul relațional logic „oricare ar fi". Elementul forall reprezintă cuantificatorul logic universal. Trebuie să fie folosit împreună cu una sau mai multe variabile dependente bvar, un element condition opțional și o afirmație care poate să fie sub forma unui element apply sau reln, dar utilizarea elementului reln este descurajată în MathML 2.0.

- **<exists />** – Operatorul relațional logic „există". Elementul exists reprezintă cuantificatorul logic de existență. Trebuie să fie folosit împreună cu una sau mai multe variabile dependente bvar, un element condition opțional și o afirmație care poate să fie sub forma unui element apply sau reln, dar utilizarea elementului reln este descurajată în MathML 2.0.
- **<abs />** – Operatorul de valoare absolută a unui număr real sau de modul al unui număr complex. Este un operator unar.
- **<conjugate />** – Conjugatul complex al unui număr complex. Este un operator unar.

Relații

Aceste funcții definesc relațiile algebrice simple dintre valori, printre care egal cu, diferit de și mărimea relativă. Se comportă așa cum vă așteptați.
Toate elementele următoare primesc atributul definitionURL, care vă permite să suprascrieți semnificația prestabilită cu o semnificație alternativă:

- **<eq />** – Relația *n*-ară „egal cu".
- **<neq />** – Relația binară „diferit de".
- **<gt />** – Relația *n*-ară „mai mare decât".
- **<lt />** – Relația *n*-ară „mai mic decât".
- **<geq />** – Relația *n*-ară „mai mare sau egal".
- **<leq />** – Relația *n*-ară „mai mic sau egal".

Relațiile și operatorii binari primesc două argumente, iar operatorii *n*-ari primesc orice număr de argumente, inclusiv zero.

Calcul

Aceste funcții sunt legate într-un fel sau altul de efectuarea unor calcule. Următoarele cinci funcții primesc atributul definitionURL, care permite suprascrierea semnificației prestabilite cu o semnificație alternativă. Restul funcțiilor din listă nu primesc acest atribut.

- **<ln />** – Operatorul logaritm natural. Acest element primește un singur argument.
- **<log />** – Operatorul logaritm cu o anumită bază. Dacă este prezent, elementul logbase specifică baza. Dacă este absent, se presupune baza 10 și este suficient un singur argument.
- **<int />** – Operatorul de integrare. Dacă sunt prezente, elementele opționale lowlimit, uplimit și bvar cuprind limita inferioară, limita superioară și variabila de integrare. În caz contrar, un element interval al unui element condiție poate să specifice domeniul de integrare. Dacă integrarea este definită, trebuie să se specifice explicit variabila de integrare.
- **<diff />** – Operatorul de derivare. Elementul bvar definește variabila față de care se derivează, iar un element opțional degree inclus în elementul bvar indică ordinul derivatei care se va calcula.
- **<partialdiff />** – Operatorul de derivare parțială pentru funcții cu mai multe variabile reale. Variabilele sunt indicate prin elemente bvar, care sunt copii ai elementului apply care le conține. Elementele bvar pot să conțină un element degree, care indică ordinul derivatei care se va calcula pentru variabila respectivă.

- **<lowlimit>** – Un calificativ care indică limita inferioară a unui element care primește calificative. Acest element conține limita inferioară, dacă este prezentă.
- **<uplimit>** – Un calificativ care indică limita superioară a unui element care primește calificative. Acest element conține limita superioară, dacă este prezentă.
- **<bvar>** – Un calificativ care indică variabila `bound` a unui element care primește calificative. Acest element conține variabila `bound`, dacă este prezentă.
- **<degree>** – Un calificativ care indică gradul unui element care primește calificative. Acest element conține gradul sau ordinul unei operații.

Elemente legate de teoria mulțimilor

Următoarele elemente exprimă relații pentru mulțimi:

- **<set>** – Construiește o mulțime. Elementul `set` este un container care construiește o mulțime de elemente din conținutul său. Elementele mulțimii pot fi definite fie prin enumerarea lor explicită, fie prin utilizarea unor elemente `bvar` și `condition` care dau una sau mai multe condiții de definiție pentru valorile componente ale mulțimii.
- **<list>** – Construiește o listă. Elementul `list` este un container care construiește o listă de elemente din conținutul său. Elementele listei pot fi definite fie prin enumerarea lor explicită, fie prin utilizarea unor elemente `bvar` și `condition` care dau una sau mai multe condiții de definiție pentru valorile componente ale listei.
  Listele diferă de mulțimi prin faptul că există o ordine explicită a elementelor. Sunt suportate două tipuri de ordine: `'lexicographic'` și `'numeric'`. Puteți specifica tipul de ordine dorit utilizând atributul `order`.

Toate elementele următoare primesc atributul `definitionURL`, care permite suprascrierea semnificației prestabilite cu o semnificație alternativă. Se comportă așa cum vă așteptați. Relațiile și operatorii binari primesc două argumente, iar operatorii *n*-ari primesc orice număr de argumente:

- **<union />** – Operatorul *n*-ar de reuniune a mulțimilor.
- **<intersect />** – Operatorul *n*-ar de intersecție a mulțimilor.
- **<in />** – Relația binară de incluziune a mulțimilor.
- **<notin />** – Relația binară de ne-incluziune a mulțimilor.
- **<subset />** – Relația *n*-ară de a fi submulțime a unei mulțimi.
- **<prsubset />** – Relația *n*-ară de a fi submulțime proprie a unei mulțimi.
- **<notsubset />** – Relația binară de a nu fi o submulțime a unei mulțimi.
- **<notprsubset />** – Relația binară de a nu fi o submulțime proprie a unei mulțimi.
- **<setdiff />** – Operatorul binar de diferență a mulțimilor.

Șiruri și serii

Următoarele funcții sunt legate de șiruri și serii și toate primesc atributul `definitionURL`, care permite suprascrierea semnificației prestabilite a elementului cu o semnificație alternativă.

- **<sum />** – Operatorul de însumare. Limitele superioară și inferioară ale sumei sunt specificate folosind `uplimit`, `lowlimit` sau printr-un element `condition` impus unor elemente `bvar`. Indicele după care se însumează este indicat printr-un element variabilă `bvar`.
- **<product />** – Operatorul de produs. Limitele superioară și inferioară ale produsului sunt specificate folosind `uplimit`, `lowlimit` sau printr-un element `condition` impus unor elemente legate. Indicele produsului este indicat printr-un element variabilă `bvar`.
- **<limit />** – Operația de aflare a limitei unui șir. Limita este exprimată prin specificarea `lowlimit` și a unei variabile `bvar` sau printr-un element `condition` impus unuia sau mai multor elemente variabile `bvar`.
- **<tendsto />** – Afirmă că o cantitate tinde spre o mărime specificată. Acest element primește un atribut `type`, care poate lua valorile `above`, `below` sau `two-sided` pentru a indica direcția din care se apropie de valoarea limită.

Trigonometrie

Elementele de trigonometrie sunt funcțiile clasice pe care oricine și le poate aminti din școala generală, sinus, cosinus și toate celelalte. Fiecare dintre elementele următoare primește atributul `definitionURL`, care permite suprascrierea semnificației prestabilite a elementului cu o semnificație alternativă.

- **<sin />** – Sinus
- **<cos />** – Cosinus
- **<tan />** – Tangentă
- **<sec />** – Secantă
- **<csc />** – Cosecantă
- **<cot />** – Cotangentă
- **<sinh />** – Sinus hiperbolic
- **<cosh />** – Cosinus hiperbolic
- **<tanh />** – Tangentă hiperbolică
- **<sech />** – Secantă hiperbolică
- **<csch />** – Cosecantă hiperbolică
- **<coth />** – Cotangentă hiperbolică
- **<arcsin />** – Arcsinus
- **<arcos />** – Arccosinus
- **<arctan />** – Arctangentă

Elementele de mai sus se referă la funcțiile trigonometrice unare obișnuite. MathML 2.0 schimbă eticheta acestor elemente de la „trigonometric“ la „classic“, dar în rest nu mai face nici o schimbare.

Statistică

Elementele de statistică implementează funcțiile statistice elementare. Fiecare primește atributul `definitionURL`, care permite suprascrierea semnificației prestabilite a elementului cu o semnificație alternativă.

- **<mean />** – Operatorul de medie statistică. Elementul `mean` returnează media unei liste de valori.
- **<sdev />** – Operatorul statistic de deviație standard. Elementul `sdev` returnează deviația standard a unei liste de valori.
- **<variance />** – Operatorul statistic de varianță. Elementul `variance` returnează variația unei liste de valori.

- <median /> – Operatorul statistic de mediană. Elementul median returnează mediana unei liste de valori.
- <mode /> – Operatorul statistic de mod sau dominantă. Elementul mode returnează modul sau dominanta unei liste de valori.
- <moment /> – Un moment statistic. Elementul moment reprezintă momente statistice. Puteți folosi elementul degree pentru n în „momentul de ordin n".

Elementele de mai sus se referă la funcțiile statistice clasice.

Algebră liniară

Aceste elemente definesc funcțiile clasice din algebra liniară:

- <vector> – Construiește un vector. Elementul vector este un container pentru vector. Elementele copil formează componentele vectorului.
- <matrix> – Construiește o matrice. Elementul matrix este un container pentru liniile matricei. Elementele matrixrow conțin elementele matricei.
- <matrixrow> – Construiește o linie a unei matrice. Elementul matrixrow este un container pentru o linie a unei matrice.
- <determinant> – Construiește determinantul unei matrice. Elementul determinant este operatorul care construiește determinantul unei matrice.
- <transpose> – Transpune o matrice. Elementul transpose este operatorul care construiește transpusa unei matrice.
- <selector> – Indice într-o listă, vector, tablou sau matrice.

  Acest element primește ID-ul unei liste, vector, tablou sau matrice , iar următoarele două argumente identifică coloana și linia. Dacă există trei argumente dar se identifică o listă sau un vector, al treilea argument este ignorat. Dacă se identifică o matrice sau un tablou dar există numai două argumente, atunci elementele tabloului sunt așezate după coloane și apoi după rânduri, ca un simplu vector, iar apoi este returnat elementul corespunzător.

Elemente de reprezentare a semnificațiilor

Aceste elemente vă permit să modificați semnificațiile unor funcții predefinite:

- <annotation> – O semnificație alternativă care nu se bazează pe XML. Elementul annotation este întotdeauna conținut într-un element semantics, dacă există. Primește un atribut de codificare, folosit pentru a identifica tipul codificării folosite într-un format de adnotare care nu este XML. Un exemplu ar putea fi <annotation encoding='TeX'> instructiuni TeX </annotation>. Dacă un browser poate sau nu să folosească această informație, depinde de aplicație.
- <semantics> – Furnizează semnificații alternative pentru o expresie MathML. Primul element este întotdeauna o construcție MathML de conținut și este redată în mod prestabilit. După prima construcție pot fi incluse oricâte formate alternative, folosind câte un element adnotare pentru fiecare format care urmează.

  Elementul semantics permite atașarea unor mecanisme de redare alternative la expresiile de conținut MathML. De exemplu, puteți include un element de prezentare MathML alternativ folosind un element inclus:

```
<annotation-xml encoding='MathML-Presentation'>
  ...
</annotation-xml>
```

Acest element primește atributul definitionURL, care permite suprascrierea semnificației prestabilite a elementului cu o semnificație alternativă.

- <annotation-xml> – O semnificație alternativă bazată pe XML. Elementul annotation-xml este întotdeauna conținut într-un element semantics, dacă există. Primește un atribut de codificare, folosit pentru a identifica tipul de codificare folosit pentru adnotarea semantică bazată pe XML. Un exemplu ar putea fi <annotation-xml encoding='OpenMath'> instructiuni OpenMath </annotation-xml>. Dacă un browser poate sau nu să folosească această informație, depinde de aplicație.

Modificările și completările la elementele de conținut propuse pentru MathML 2.0

MathML 2.0 este încă în dezvoltare, urmând să apară încă cel puțin două schițe de lucru înainte de a avansa la statutul de recomandare completă. Oricare sau toate modificările înșirate pe scurt în cele ce urmează pot fi eliminate, adoptate sau schimbate complet până la finalizarea standardului.

<reln> este suportat, dar în locul său se recomandă <apply>, care suportă acum și elemente relaționale.

Funcțiile trigonometrice au fost redenumite funcții elementare clasice.

S-au adăugat următoarele elemente:

- <csymbol> – Definește un simbol matematic.

  Elementul csymbol folosește atributul definitionURL pentru a indica definiția matematică a noului simbol. În plus, folosește elementul de codificare pentru a identifica aplicația matematică folosită în definirea noului simbol.
- <arg> – Element nou, aritmetic, algebric sau logic.
- <real> – Element nou, aritmetic, algebric sau logic.
- <imaginary> – Element nou, aritmetic, algebric sau logic.
- <equivalent> – Element relațional nou.
- <approx> – Element relațional nou.
- <divergence> – Element nou de calcul și calcul vectorial.
- <grad> – Element nou de calcul și calcul vectorial.
- <curl> – Element nou de calcul și calcul vectorial.
- <laplacian> – Element nou de calcul și calcul vectorial.
- <size> – Element nou pentru mulțimi.
- <vectorproduct> – Element nou de algebră liniară.
- <scalarproduct> – Element nou de algebră liniară.
- <outerproduct> – Element nou de algebră liniară.

Definițiile elementelor de mai sus nu au fost încă formalizate în momentul scrierii acestei cărți.

Elementul de interfață MathML

Nivelul cel mai înalt, sau elementul de interfață pentru orice conținut MathML este elementul math. Acesta este echivalentul oricărui element XML de cel mai înalt nivel și este

o eroare să includeți un element math în alt element math. Deoarece MathML a fost creat pentru a fi inclus în documentele HTML sau XML, este de asemenea o eroare să includeți instrucțiuni MathML într-un document fără să le încadrați cu etichete <math>.
În operațiile de tăiere și lipire care implică expresii MathML, elementul math care le încadrează poate să nu fie inclus, deci agentul utilizator trebuie să asigure aceste etichete dacă lipsesc. Analog, dacă o operație de tăiere și lipire inserează o expresie MathML completă, cu tot cu elemente math, într-o expresie MathML existentă, la rândul ei completă, elementele math incluse trebuie eliminate.
Atributele elementului math sunt:

class='value' – Furnizat pentru compatibilitate cu foile de stil.

id='value' – Furnizat pentru compatibilitate cu foile de stil.

style='value' – Furnizat pentru compatibilitate cu foile de stil.

macros='URL URL ...' – Atributul macros asigură o modalitate de a indica fișiere de macrodefiniții externe. În momentul de față, macroinstrucțiunile nu fac parte din specificația MathML, dar se crede că în viitor, prin intermediul transformărilor XSL (http://www.w3.org/TR/xslt), vor fi incluse numeroase utilizări ale macroinstrucțiunilor în MathML. Totuși, atributul macros este furnizat pentru a face posibilă dezvoltarea viitoare a unor mecanisme de macroinstrucțiuni eficiente, specifice MathML.

mode='display|inline' – Atributul mode specifică dacă expresia MathML inclusă trebuie redată într-un stil de afișare sau într-un stil in-line. Valoarea prestabilită este mode='inline'. Utilizarea acestui atribut este descurajată în favoarea proprietății standard CSS2 display.

Sintaxă sau prezentare

Am discutat deja în acest capitol cele două tipuri de bază ale elementelor MathML: prezentare și conținut. Deși, în mod ideal, motoarele de redare MathML ar trebui să înțeleagă conținutul și să îl afișeze corect în funcție de semnificația sa, realitatea cercetării și dezvoltării matematice este că întotdeauna se vor inventa notații noi pentru noțiuni noi, și chiar notațiile existente pot fi folosite în contexte diferite pentru a exprima înțelesuri diferite. Astfel, x^2 poate să însemne „x la pătrat" sau „a doua componentă a vectorului x".
Redarea semnificației matematice depinde și de context. Notațiile folosite de matematicieni pentru a exprima factorialul lui x diferă în fiecare situație în funcție de ce vor să facă cu informația. Un motor de redare nu știe dacă trebuie să fie $x!$, notația cea mai obișnuită, $\Gamma(x+1)$, funcția gama echivalentă, sau $(1)_x$, simbolul Pochhammer pentru același concept matematic.
Marcajul de prezentare este necesar pentru a spune motorului de redare cum trebuie să fie redată instrucțiunea MathML <apply><factorial/><ci>x</ci></apply>, în afară de cazul în care redarea se întâmplă să corespundă notației prestabilite specificate în dicționarul de operatori.

Combinarea directă a prezentării și conținutului

Există mai multe motive pentru care ați putea dori să includeți elemente de prezentare între elementele de conținut sau viceversa. Primul ar fi să permiteți aplicarea unei formatări speciale unui element ci, astfel:


```
<ci><mi fontweight='bold'>x</mi></ci>
```

Deoarece elementul ci nu permite un atribut fontweight, acesta este un mod simplu de a reda variabila cu aldine.
Și situația inversă poate să fie adevărată. Într-un document orientat spre prezentare, poate fi recomandat să folosiți etichete de conținut pentru a permite, de exemplu, persoanelor care folosesc browsere audio să distingă ușor o ridicare la putere de o componentă a unui vector, folosind setările browserului:

```
<mrow>
  <apply>
    <power/>
    <ci>x</ci><cn>2</cn>
  </apply>
  <mo>+</mo>
  <msup>
    <mi>x</mi><mn>2</mn>
  </msup>
</mrow>
```

Deoarece afișarea prestabilită a unui număr ridicat la o putere este aproape sigur identică cu cea a unui indice superior oarecare, afișarea va fi normală într-un browser vizual. În schimb, dacă se folosesc ambele scheme de etichetare, un browser audio le va distinge utilizând intonații diferite.

Utilizarea elementelor de prezentare în marcajul de conținut

Pentru a evita problemele de descifrare a înțelesului când se folosesc elemente de prezentare în marcajul de conținut, utilizarea elementelor de prezentare este interzisă, cu următoarele excepții:

- În interiorul elementelor simbolice ci sau cn
- În interiorul elementului fn
- În interiorul elementului semantics

În particular, elementul semantics a fost conceput pentru a permite vizualizări arbitrare sau pentru a atașa reprezentări semantice la elementele de conținut.

Utilizarea elementelor de conținut în marcajul de prezentare

Singurele elemente de conținut permise în marcajul de prezentare sunt etichetele container precum <ci>, <cn>, <apply>, <fn>, <lambda>, <reln>, <interval>, <list>, <matrix>, <matrixrow>, <set>, <vector> și <declare>.
Înțelesul celorlalte elemente depinde în parte de context, deci sunt interzise în acest context.

Modificările și adăugirile la MathML propuse pentru combinarea prezentării și conținutului

Atributul xref poate să fie combinat cu atributul id pentru a permite referirea fără ambiguități între elementele de prezentare și cele de conținut. În utilizare, veți identifica pur și simplu fiecare element de prezentare cu un ID XML, iar apoi veți adăuga un xref la elementul de prezentare corespunzător în fiecare element de sintaxă.

Combinarea matematicii și textului în MathML

MathML a fost creat pentru a fi folosit în fragmente matematice incluse în documente mai mari, astfel încât elementul `math` de cel mai înalt nivel este o componentă-cheie a strategiei complete de redare. Deoarece elementul `math` este stabil, un browser trebuie numai să știe că tot ce este în interior trebuie redat în condiții speciale sau ignorat, fără altă variantă intermediară. Din nefericire, este improbabil să se întâmple așa în momentul de față, deoarece browserele moștenite încearcă adesea să redea totul.

Deoarece fragmentele MathML (și toate documentele sau fragmentele de documente XML) conțin date text expuse pentru multe browsere, trebuie să se asigure un mijloc prin care să se evite redarea nepotrivită a datelor expuse. În lumea HTML va exista întotdeauna această tensiune dintre noile facilități și cele vechi, iar imposibilitatea asigurării suportului (sau chiar a ignorării) noilor capabilități în browserele mai vechi indică faptul că unele lucruri pur și simplu nu pot fi făcute. Deși orice browser HTML ar trebui să ignore etichetele pe care nu le cunoaște, singura modalitate de a ascunde conținutul de text expus este să îl eliminați complet din document.

Asta înseamnă că va trebui să creați documente externe care să conțină caracteristicile noi și să le legați prin abilități care nu sunt suportate în browserele mai vechi. Noul mecanism de legare XLink este un bun exemplu. Este de presupus că un browser care suportă XLink suportă și capacitatea de a ignora conținutul expus care este înconjurat de etichete necunoscute. Singura alternativă este să folosiți limbaje de script sau facilități ale serverului pentru a furniza conținutul dinamic, în funcție de versiunea și nivelul de revizuire al browserului. În unele circumstanțe pot fi necesare ambele strategii. Figura 24.2 ilustrează această problemă cu un exemplu MathML, pe care apoi îl puteți vedea afișat corect în Figura 24.3 din secțiunea următoare. Ca să fim sinceri, afișarea este plină de nimicuri iritante. Textul afișat în Figura 24.2 *ar trebui* să fie invizibil, deoarece este o porțiune din codul care generează imaginea tridimensională din Figura 24.3. Aceasta este o problemă de proiectare a limbajului, și nu o eroare a producătorului browserului. Ideal, ați putea ocoli problema prin etichete obiect HTML sau XHTML imbricate, dar multe browsere nu suportă nici această sintaxă.

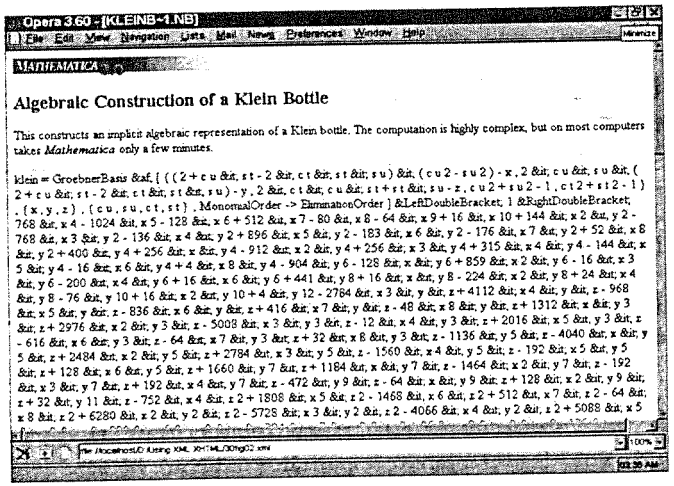


FIGURA 24.2
Browserul Opera vă prezintă conținutul text MathML expus.

Deși proiectanții noilor limbaje presupun că browserele vechi vor fi înlocuite curând, cert este că majoritatea utilizatorilor nu își modernizează browserul decât dacă își cumpără un calculator nou care este furnizat cu un browser nou. Programatorii tind să „adopte devreme" și presupun că și alții sunt sau ar trebui să fie la fel. Asta nu se va întâmpla niciodată. Ca proiectant Web, trebuie să vă obișnuiți cu această idee pentru totdeauna, cu excepția cazului în care sunteți într-o rețea internă cu un control strict asupra configurației și vi se poate garanta versiunea și nivelul de revizie pentru fiecare browser din rețea.

Tehnicile prin care se asigură suportul mai multor browsere nu fac subiectul acestei cărți, dar sunt discutate pe larg în aproape orice carte despre proiectarea și codarea HTML. În general, codul client-side vă încetinește browserul, iar codul server-side vă încetinește serverul. Este posibil să aveți puțin din ce este mai bun în fiecare lume, distribuind judicios programarea server-side și cea client-side, prin încorporarea suportului de bază pentru browserele cele mai obișnuite pe situl dumneavoastră și crearea pe server a paginilor particularizate, pentru a suporta browserele mai rare.

Această abordare are dezavantajul că anulează majoritatea schemelor de cache, deci majoritatea reușitelor vor trece spre server. Dar nimic nu este perfect, nu?

Instrumente MathML

Instrumentul MathML cel mai ușor de găsit este browserul W3C Amaya. În plus, acesta este gratuit. Din nefericire, domeniul de caracteristici MathML pe care le suportă este limitat. Prin acest fapt, reflectă majoritatea instrumentelor de afișare și editare. Mozilla va asigura suportul MathML și, de fapt, etichetele de prezentare există deja în versiunile anterioare lansării pe piață.

Dacă intrați însă în domeniul instrumentelor profesionale, posibilitățile devin mult mai ispititoare. Figura 24.3 vă prezintă un exemplu extras din programul Mathematica al Wolfram Research, la `http://www.wolfram.com/`, care ilustrează capacitățile extinse de reprezentare grafică ale acestui pachet. Mathematica este orientat spre explorarea științifică, precum și spre publicare, și dispune de o bibliotecă extinsă de aplicații gata pregătite pentru domenii științifice specializate.

Listingul 24.1 vă prezintă codul MathML al unei formule complexe, exportată din mediul software Mathematica. Deși acest listing poate să vă intimideze la început, chiar după o simplă privire vă veți da seama că este alcătuit din elemente MathML destul de obișnuite, în majoritate `mrow`, `mo` și `mi`. Redarea ecuației manual poate fi destul de plictisitoare, dar în mod sigur nu este imposibilă pentru orice persoană care ar putea folosi MathML în plan profesional. Utilizarea TeX și LaTeX pentru a tipări matematică este cel puțin la fel de complexă și consumatoare de timp, iar oamenii de știință o fac zilnic.

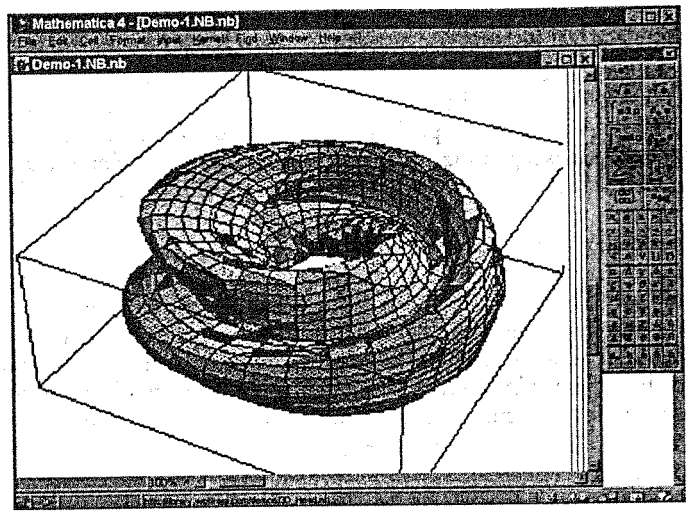


FIGURA 24.3
Graficul suprafeței generate de formulă, afișat aici cu Mathematica 4.0.

Listingul 24.1 O formulă MathML exportată din Mathematica

```
<math>
  <mrow>
    <mi>klein</mi>
    <mo>=</mo>
    <mrow>
      <mrow>
        <mi>GroebnerBasis</mi>
        <mo>&af;</mo>
        <mrow>
          <mo>[</mo>
          <mrow>
            <mrow>
              <mo>{</mo>
              <mrow>
                <mrow>
                  <mrow>
                    <mrow>
                      <mo>(</mo>
                      <mrow>
                        <mn>2</mn>
                        <mo>+</mo>
                        <mrow>
                          <msub>
                            <mi>c</mi>
                            <mi>u</mi>
                          </msub>
                          <mo>&it;</mo>
                          <msub>
                            <mi>s</mi>
                            <mi>t</mi>
                          </msub>
                        </mrow>
                      </mrow>
                      <mo>-</mo>
                      <mrow>
                        <mn>2</mn>
                        <mo>&it;</mo>
                        <msub>
                          <mi>c</mi>
                          <mi>t</mi>
                        </msub>
                        <mo>&it;</mo>
                        <msub>
                          <mi>s</mi>
                          <mi>t</mi>
                        </msub>
                        <mo>&it;</mo>
                        <msub>
                          <mi>s</mi>
                          <mi>u</mi>
                        </msub>
                      </mrow>
                    </mrow>
                    <mo>)</mo>
                  </mrow>
                  <mo>&it;</mo>
                  <mrow>
                    <mo>(</mo>
                    <mrow>
                      <msubsup>
                        <mi>c</mi>
                        <mi>u</mi>
                        <mn>2</mn>
                      </msubsup>
                      <mo>-</mo>
                      <msubsup>
                        <mi>s</mi>
                        <mi>u</mi>
                        <mn>2</mn>
                      </msubsup>
                    </mrow>
                    <mo>)</mo>
                  </mrow>
                </mrow>
                <mo>-</mo>
                <mi>x</mi>
              </mrow>
              <mo>,</mo>
              <mrow>
                <mrow>
                  <mn>2</mn>
                  <mo>&it;</mo>
                  <msub>
                    <mi>c</mi>
```

Listingul 24.1 (continuare)

```
            <mi>u</mi>
          </msub>
          <mo>&it;</mo>
          <msub>
            <mi>s</mi>
            <mi>u</mi>
          </msub>
          <mo>&it;</mo>
          <mrow>
            <mo>(</mo>
            <mrow>
              <mn>2</mn>
              <mo>+</mo>
              <mrow>
                    <msub>
                      <mi>c</mi>
                      <mi>u</mi>
                    </msub>
                    <mo>&it;</mo>
                    <msub>
                      <mi>s</mi>
                      <mi>t</mi>
                    </msub>
              </mrow>
              <mo>-</mo>
              <mrow>
                <mn>2</mn>
                <mo>&it;</mo>
                <msub>
                  <mi>c</mi>
                  <mi>t</mi>
                </msub>
                <mo>&it;</mo>
                <msub>
                  <mi>s</mi>
                  <mi>t</mi>
                </msub>
                <mo>&it;</mo>
                <msub>
                  <mi>s</mi>
                  <mi>u</mi>
                </msub>
              </mrow>
            </mrow>
            <mo>)</mo>
          </mrow>
        </mrow>
        <mo>-</mo>
        <mi>y</mi>
      </mrow>
      <mo>,</mo>
      <mrow>
        <mrow>
          <mn>2</mn>
          <mo>&it;</mo>
          <msub>
            <mi>c</mi>
            <mi>t</mi>
          </msub>
          <mo>&it;</mo>
          <msub>
            <mi>c</mi>
            <mi>u</mi>
          </msub>
          <mo>&it;</mo>
          <msub>
            <mi>s</mi>
            <mi>t</mi>
          </msub>
        </mrow>
        <mo>+</mo>
        <mrow>
          <msub>
            <mi>s</mi>
            <mi>t</mi>
          </msub>
          <mo>&it;</mo>
          <msub>
            <mi>s</mi>
            <mi>u</mi>
          </msub>
        </mrow>
        <mo>-</mo>
        <mi>z</mi>
      </mrow>
      <mo>,</mo>
      <mrow>
        <msubsup>
          <mi>c</mi>
          <mi>u</mi>
          <mn>2</mn>
        </msubsup>
        <mo>+</mo>
        <msubsup>
          <mi>s</mi>
          <mi>u</mi>
          <mn>2</mn>
        </msubsup>
        <mo>-</mo>
        <mn>1</mn>
      </mrow>
      <mo>,</mo>
      <mrow>
        <msubsup>
          <mi>c</mi>
```

Listingul 24.1 (continuare)

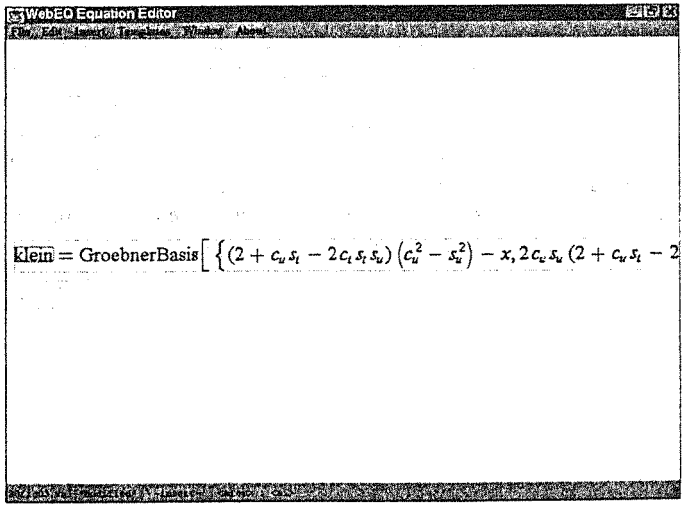
```
            <mi>t</mi>
            <mn>2</mn>
          </msubsup>
          <mo>+</mo>
          <msubsup>
            <mi>s</mi>
            <mi>t</mi>
            <mn>2</mn>
          </msubsup>
          <mo>-</mo>
          <mn>1</mn>
        </mrow>
      </mrow>
      <mo>}</mo>
    </mrow>
    <mo>,</mo>
    <mrow>
      <mo>{</mo>
      <mrow>
        <mi>x</mi>
        <mo>,</mo>
        <mi>y</mi>
        <mo>,</mo>
        <mi>z</mi>
      </mrow>
      <mo>}</mo>
    </mrow>
    <mo>,</mo>
    <mrow>
      <mo>{</mo>
      <mrow>
        <msub>
          <mi>c</mi>
          <mi>u</mi>
        </msub>
        <mo>,</mo>
        <msub>
          <mi>s</mi>
          <mi>u</mi>
        </msub>
        <mo>,</mo>
        <msub>
          <mi>c</mi>
          <mi>t</mi>
        </msub>
        <mo>,</mo>
        <msub>
          <mi>s</mi>
          <mi>t</mi>
        </msub>
      < mrow>
      <mo>}</mo>
```



```
        </mrow>
        <mo>,</mo>
        <mrow>
          <mi>MonomialOrder</mi>
          <mo>-></mo>
          <mi>EliminationOrder</mi>
        </mrow>
      </mrow>
      <mo>]</mo>
    </mrow>
  </mrow>
  <mo>&LeftDoubleBracket;</mo>
  <mn>1</mn>
  <mo>&RightDoubleBracket;</mo>
 </mrow>
</mrow>
</math>
```

Figura 24.4 vă prezintă aceeaşi formulă importată în editorul de ecuaţii WebEQ, care este un instrument specializat de la Geometry Technologies (`http://www.webeq.com/`) conceput în special pentru MathML. Aceasta demonstrează că puteţi să transmiteţi cu uşurinţă lucrările pe care le-aţi creat pe o platformă unui coleg care lucrează cu un instrument complet diferit, prin mediul universal de transfer reprezentat de MathML.

FIGURA 24.4
Formula creată în Mathematica este afişată cu WebEQ



Alte câteva instrumente sunt Maple, de la Waterloo Maple, la `http://www.maplesoft.com/`, MathCAD, la `http://www.mathcad.com/`, Macsyma, de la Macsyma, la `http://www.macsyma.com/`, şi multe altele. Sistemele de calcul algebric sunt disponibile peste tot, multe fiind gratuite sau foarte ieftine, iar majoritatea sunt de foarte bună calitate. În nici un caz acestea nu suportă acum MathML, dar suportul va fi disponibil în viitorul apropiat pentru majoritatea sistemelor de calcul algebric.

CML şi alte aplicaţii ştiinţifice ale XML

CML, limbajul de marcare pentru chimie (The Chemical Markup Language) este o altă aplicaţie bazată pe XML, creată pentru cercetătorii în activitate. Acesta le permite să definească exact structura moleculelor şi să vizualizeze rezultatele folosind editoare/browsere speciale. Ca şi MathML, CML este proiectat pentru a conlucra cu instrumente puternice şi pentru a servi ca mediu de transfer între diferite medii de editare şi afişare pentru chimie.

Această secţiune vă prezintă un singur exemplu CML pentru a arăta o parte din puterea aplicaţiilor ştiinţifice ale XML în general. CML este pur şi simplu un exemplu din acest domeniu foarte larg, şi nu este cu nimic mai special decât alte aplicaţii. În particular, nu vom explica în această secţiune ce înseamnă elementele CML sau cum sunt asamblate, deoarece, în mare, ştiţi deja cum funcţionează XML, şi toate vocabularele XML ascultă de aceleaşi reguli.

Situl CML, la `http://www.xml-cml.org/`, conţine informaţii despre software-ul care există, inclusiv despre Jumbo, browserul CML, dar este puţin învechit. Situl lui Peter Murray-Rust, la `http://www.ch.ic.ac.uk/omf/cml/`, conţine de asemenea materiale interesante, deşi nici acesta nu a fost actualizat de ceva vreme. Peter Murray-Rust are intenţia să actualizeze CML şi lucrează la îmbunătăţirea şi extinderea influenţei acestuia în comunitatea ştiinţifică.

Un alt software ştiinţific similar este MOLMOL, programul de analiză moleculară şi afişare a moleculelor de la `http://www.mol.biol.ethz.ch/wuthrich/software/molmol/` şi mai există sute de alte programe, majoritatea dintre acestea însă nu pot comunica, la momentul de faţă, cu celelalte. Un alt sit cu legături spre numeroase pachete de software pentru comunităţile ştiinţifice de biologie şi chimie este `http://www.infochem.co.uk/software/bio/index.htm`.

Un exemplu al CML în plină acţiune este disponibil pe situl domnului Murray-Rust la `http://www.ch.ic.ac.uk/omf/cml/doc/examples/csd.html` şi este folosit aici cu permisiunea autorului, care a depus o muncă extenuantă pentru comunitatea XML în multe direcţii, atât prin limbajul său CML cât şi prin serviciile inestimabile aduse pentru XML-DEV.

Figura 24.5 vă prezintă o imagine moleculară generată de un document CML.

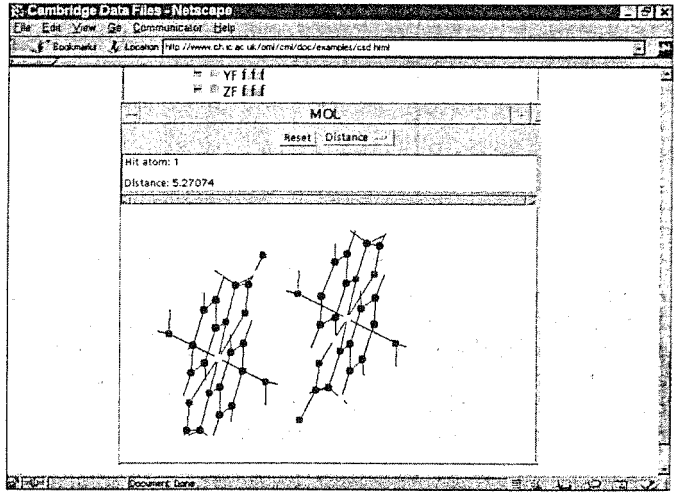


FIGURA 24.5
Browserul CML Jumbo afişează molecula descrisă de codul CML.

Figura 24.5 a fost generată de codul CML din Listingul 24.2, demonstrând cât de uşor pot browserele specializate să facă accesibile documente XML foarte complexe, îmbunătăţind înţelegerea şi vizualizarea datelor ştiinţifice, în acest caz structura moleculelor.

Listingul 24.2 Un document CML primit de la autorul instrumentului

```
<?XML VERSION='1.0'?>
<!DOCTYPE CML PUBLIC '-//CML//DTD CML/EN'>
  <CML>
    <MOL>
      <BIB TITLE='Citation'>
        <PERSON>
          <XVAR BUILTIN='LASTNAME'>
            PROUT
          </XVAR>
          <XVAR BUILTIN='INITIALS'>
            C.K
          </XVAR>
        </PERSON>
        <PERSON>
          <XVAR BUILTIN='LASTNAME'>
            ARMSTRONG
          </XVAR>
          <XVAR BUILTIN='INITIALS'>
            R.A
          </XVAR>
        </PERSON>
        <PERSON>
          <XVAR BUILTIN='LASTNAME'>
            CARRUTHERS
          </XVAR>
          <XVAR BUILTIN='INITIALS'>
            J.R
          </XVAR>
        </PERSON>
        <PERSON>
          <XVAR BUILTIN='LASTNAME'>
            FORREST
          </XVAR>
          <XVAR BUILTIN='INITIALS'>
            J.G
          </XVAR>
        </PERSON>
        <PERSON>
          <XVAR BUILTIN='LASTNAME'>
            MURRAY-RUST
          </XVAR>
          <XVAR BUILTIN='INITIALS'>
            P
          </XVAR>
        </PERSON>
        <PERSON>
```

Listingul 24.2 (continuare)

```
        <XVAR BUILTIN='LASTNAME'>
          ROSSOTTI
        </XVAR>
        <XVAR BUILTIN='INITIALS'>
          F.J.C
        </XVAR>
      </PERSON>
      <XVAR BUILTIN='JOUR'>
        J.CHEM.SOC.A
      </XVAR>
    </BIB>
  <FORMULA CONVENTION='CCDC'>
    <XVAR BUILTIN='STOICH'>
      C16 H18 CU1 O8
    </XVAR>
    <ARRAY BUILTIN='ELSYM'>
      CU 'O ' 'O ' 'O ' 'O ' 'O ' 'C ' 'C ' 'C ' 'C ' 'C ' 'C '
      'C ' 'C ' 'C ' 'C ' 'O ' 'O ' 'C ' 'C ' 'C ' 'C ' 'C ' 'C '
    </ARRAY>
    <ARRAY BUILTIN='NONH'>
      6 3 3 2 2 1 1 3 2 3 2 3 3 2 2 2 2 1 1 2 2 2 2 2 2
    </ARRAY>
    <ARRAY BUILTIN='NUMH'>
      0 0 0 0 0 2 2 0 2 0 2 0 0 1 1 1 1 0 0 1 1 1 1 1 1
    </ARRAY>
    <ARRAY BUILTIN='FORMCHARGE'>
      0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    </ARRAY>
    <ARRAY BUILTIN='ATID1'>
      1 1 1 1 1 1 2 2 3 3 4 5 8 8 9 10 10 11 12 13 14 15 16 17
    </ARRAY>
    <ARRAY BUILTIN='ATID2'>
      2 3 4 5 6 7 8 9 10 11 12 13 14 15 12 16 17 13 18 19 20
      21 22 23 24 24 25 25
    </ARRAY>
    <ARRAY BUILTIN='ORDER'>
      -1 -1 -1 -1 1 1 1 -1 1 -1 -1 -1 -5 -5 -1 -5 -5 -1 2 2
      -5 -5 -5 -5 -5 -5 -5 -5
    </ARRAY>
  </FORMULA>
  <CRYST>
    <XVAR BUILTIN='ACELL' TYPE='FLOAT' UNITS='ANGSTROM'>
      16.26
    </XVAR>
    <XVAR BUILTIN='BCELL' TYPE='FLOAT' UNITS='ANGSTROM'>
      7.28
    </XVAR>
    <XVAR BUILTIN='CCELL' TYPE='FLOAT' UNITS='ANGSTROM'>
      23.99
    </XVAR>
    <XVAR BUILTIN='ALPHA' TYPE='FLOAT' UNITS='DEGREES'>
      90
```



```
    </XVAR>
    <XVAR BUILTIN='BETA' TYPE='FLOAT' UNITS='DEGREES'>
      90
    </XVAR>
    <XVAR BUILTIN='GAMMA' TYPE='FLOAT' UNITS='DEGREES'>
      110.4
    </XVAR>
    <XVAR BUILTIN='RFACTOR' TYPE='FLOAT'>
      0.113
    </XVAR>
    <XVAR BUILTIN='COMMENT'>
      Atom H16 removed from ATOM-list, for suspected coordinate error(s)
    </XVAR>
    <XVAR BUILTIN='COMMENT'>
      C15 z -0.1763, not 0.1763 C5 z 0.1742, not 0.1762 C7 z 0.2151,
      not 0.2171 C14 z -0.138, not -0.135 C18 z -0.1739, not -0.1729
      O13 y -0.1588, not 0.1588. Phenyl ring defined by C23-C28 have been
      intercanged in figure 2, the correct values are C23-C24 1.38,C24-C25
      1.46, C25-C26 1.38, C26-C27 1.35, C27-C28 1.43 and C28-C231.40
    </XVAR>
  </CRYST>
  <SYMMETRY TITLE='Symmetry'>
    <ARRAY BUILTIN='SYMOP' ROWS='4' COLUMNS='3' TYPE='FLOAT'
STRUCT='RECTANGULAR'>
      1 0 0 0 0 1 0 0 0 0 1 0
    </ARRAY>
    <ARRAY BUILTIN='SYMOP' ROWS='4' COLUMNS='3' TYPE='FLOAT'
STRUCT='RECTANGULAR'>
      -1 0 0 0.5 0 -1 0 0.5 0 0 1 0.5
    </ARRAY>
  </SYMMETRY>
  <ATOMS>
    <ARRAY BUILTIN='ELSYM'>
      CU CU C C C C C C C C C C C C C C C C C C C C C C C H H H
      H H H H H H H H H H H H H H H H H H H H H H H H O O O O O O O
      O O O O O O O O C C C H H O H H C C C H C H C H H H
    </ARRAY>
    <ARRAY BUILTIN='ATID'>
      CU1 CU2 C1 C11 C12 C13 C14 C15 C16 C17 C18 C2 C21 C22 C23 C24
      C25 C26 C27 C28 C3 C4 C5 C6 C7 C8 H1 H11 H12 H13 H14 H15 H17
      H18 H19 H2 H21 H22 H23 H24 H25 H26 H27 H28 H29 H3 H4 H5 H6 H7
      H8 H9 O1 O11 O12 O13 O14 O2 O21 O22 O23 O24 O3 O4 O1B O3B O4B
      C1B C2B C3B H3B H9B O2B H1B H2B C4B C8B C5B H4B C7B H8B C6B H5B
      H7B H6B
    </ARRAY>
    <ARRAY BUILTIN='XF' TYPE='FLOAT'>
      0 0.3292 -0.053 0.3892 0.324 0.237 0.2049 0.1403 0.1165 0.1507
      0.2112 0.0131 0.2774 0.3414 0.4266 0.4544 0.5173 0.5431 0.5154 .
      0.4556 0.0961 0.1246 0.1858 0.211 0.182 0.1237 0.0691 0.3569
      0.2702 0.2334 0.2277 0.1126 0.1303 0.2363 0.1759 -0.0163 0.3955
      0.3087 0.4271 0.43 0.5429 0.5469 0.5401 0.4333 0.4776 -0.0954
      0.1008 0.2114 0.2553 0.2038 0.1013 -0.152 -0.0681 0.4004 0.4312
      0.2922 0.2316 -0.0929 0.2626 0.2369 0.3725 0.4251 0.0414 -0.0968
      0.0681 -0.0414 0.0968 0.053 -0.0131 -0.0961 0.0954 0.152 0.0929
```

Listingul 24.2 (continuare)

```
        -0.0691 0.0163 -0.1246 -0.1237 -0.1858 -0.1008 -0.182 -0.1013
        -0.211 -0.2114 -0.2038 -0.2553
      </ARRAY>
      <ARRAY BUILTIN='YF' TYPE='FLOAT'>
        0 0.0344 0.3147 -0.2747 -0.3182 -0.1653 -0.0155 -0.0114 -0.1569
        -0.307 -0.315 0.344 0.3453 0.3634 0.192 0.0341 0.015 0.151 0.306
        0.331 0.1754 0.0196 0.0056 0.1478 0.304 0.3214 0.4738 -0.3337 -0.45
        -0.3034 0.1028 0.1031 -0.4211 -0.4338 -0.2175 0.3622 0.497 0.3636
        0.3765 -0.0762 -0.1041 0.1341 0.414 0.4542 0.2929 -0.3382 -0.0962
        -0.1127 0.1365 0.4148 0.4448 -0.2505 0.1738 -0.133 -0.387 -0.1588
        -0.1673 0.4345 0.2098 0.4648 0.2034 0.2392 0.1788 -0.2018 -0.1738
        -0.1788 0.2018 -0.3147 -0.344 -0.1754 0.3382 0.2505 -0.4345 -0.4738
        -0.3622 -0.0196 -0.3214 -0.0056 0.0962 -0.304 -0.4448 -0.1478 0.1127
        -0.4148 -0.1365
      </ARRAY>
      <ARRAY BUILTIN='ZF' TYPE='FLOAT'>
        0 0.0004 0.0397 -0.0325 -0.0817 -0.1327 -0.138 -0.1763 -0.2162
        -0.2146 -0.1739 0.0867 0.0426 0.0897 0.132 0.1302 0.1709 0.2134
        0.2149 0.1744 0.1344 0.135 0.1742 0.2175 0.2151 0.1757 0.0782 -0.1192
        -0.0735 0.0397 -0.1048 -0.1755 -0.2461 -0.1726 0.0169 0.1252 0.085
        0.1285 -0.0349 0.0981 0.1678 0.2459 0.2477 0.1769 -0.0149 0.0376
        0.1037 0.1752 0.2504 0.2493 0.1757 0.0159 0.0051 0.0005 -0.0272
        -0.0889 0.0389 0.0363 0.0055 0.0415 0.089 -0.0393 0.0904 0.0384
        -0.0051 -0.0904 -0.0384 -0.0397 -0.0867 -0.1344 -0.0376 -0.0159
        -0.0363 -0.0782 -0.1252 -0.135 -0.1757 -0.1742 -0.1037 -0.2151
        -0.1757 -0.2175 -0.1752 -0.2493 -0.2504
      </ARRAY>
    </ATOMS>
  </MX>
</CML>
```

Detaliile complete ale CML sunt mult prea complexe pentru a le discuta aici, dar dacă sunteți interesat de acest domeniu, vă recomand să continuați prin studiu individual. De asemenea, JUMBO și CML nu sunt totul în știința moleculară. Există multe aplicații gratuite care facilitează activități speciale, cum ar fi „potrivirea" – descoperirea prin simulare a modului în care se „potrivesc" moleculele – care determină acțiunile posibile ale moleculei fără o sinteză dificilă în laborator, fără testare și altele.

Pachete științifice similare sunt sau vor fi disponibile în multe alte domenii o dată cu acceptarea XML în comunitatea științifică. Deja sunt în dezvoltare AML, Astronomical Markup Language – limbajul de marcare pentru astronomie, AIML, Astronomical Instrument Markup Language – limbajul de marcare pentru instrumente astronomice, CML, Instrument Markup Language – limbajul de marcare pentru instrumente și multe altele.

Studii de caz

MathML și CML au multe asemănări, în ciuda aspectului lor diferit, prin faptul că domeniile la care se adresează sunt foarte specializate, sunt studiile unor profesioniști în

știință, și nu componente ale informațiilor generale pe care oricine ar trebui să le știe. Dacă aveți nevoie de aceste facilități, atunci aveți într-adevăr foarte mare nevoie de ele, deoarece comunicarea între oamenii de știință stă la baza cercetării moderne.

Aceste limbaje au în comun abilitatea de a comunica și de a transfera fără întreruperi informațiile pe Web folosind un limbaj de aplicație bazat pe XML. Ca mediu de transfer, de altfel și ca format de stocare, documentele bazate pe XML au avantajul că sunt lizibile pentru om și sunt foarte robuste. Într-un mediu în care o singură modernizare a softului poate duce la căderea formatelor de date binare, MathML și celelalte limbaje bazate pe XML își păstrează abilitatea de a decoda datele cu ușurință fără a ști dinainte formatul. În cazul MathML, un matematician care se uită peste codul din Listingul 24.1 ar putea să deducă ușor semnificația majorității etichetelor, deoarece se supun regulilor matematice obișnuite și sunt în cea mai mare parte bazate pe engleză, limbă cunoscută și folosită pe larg în cercurile științifice. Chiar și într-un viitor îndepărtat, în care engleza va fi o relicvă uitată din istoria antică, regularitatea etichetelor și universalitatea matematicii vor asigura deducerea întregii structuri a XML, și a MathML în particular, din câteva exemple simple ca acesta.

Accesibilitate

CAPITOLUL

25

În acest capitol

Privire generală asupra accesibilității pe Web
Accesibilitatea din punct de vedere uman
Accesibilitatea din punct de vedere economic
Accesibilitatea din punct de vedere legal
Tehnici de afișare pentru vedere slabă
Să facem față tehnicilor Braille
Utilizarea foilor de stil auditive și a browserelor audio
Realizarea unui document XML accesibil
Rezolvarea problemelor de accesibilitate

Privire generală asupra accesibilității pe Web

În minunata lume nouă a XML, accesibilitatea, așa cum este definită în Inițiativa pentru accesibilitate Web (WAI – Web Accessibility Initiative) a W3C, ar trebui să fie universală. Chiar scopul XML este să facă documentele accesibile într-un format independent de platformă, care să fie „rezistent la evoluție", bazat pe text și pe etichete descriptive, astfel încât oricine se uită la fișier să își poată da seama ce este documentul și din ce elemente este alcătuit. WAI a fost proiectată pentru a extinde această transparență pentru oricine, astfel încât browserele, instrumentele de creare și întreaga experiență Web să fie la fel de plăcute și de utile și pentru persoanele cu handicap de vedere, motor sau cognitiv. Dar asta nu înseamnă că ați scăpat de bătăi de cap. Este greu de presupus că lumea va avea acces universal în perioada imediat următoare la browsere apte pentru XML. Deci o parte a muncii dumneavoastră va fi să vă gândiți la modalitățile prin care conținutul dumneavoastră XML devine vizibil pentru un număr cât mai mare de persoane care nu au un suport XML direct.

Sugestie de la LeeAnne.com Words to weave by

Unul dintre neajunsurile dezvoltării de software este tratarea formatelor de fișiere binare incompatibile. Adesea, formatul a evoluat până la punctul în care este imposibil de recunoscut și, deoarece este binar, este aproape imposibil de înțeles fără o analiză exhaustivă a unui număr mare de tipuri de înregistrări. XML este proiectat pentru a rezolva această problemă etichetând în mod explicit fiecare câmp, astfel încât chiar și peste cinci mii de ani un arheolog va putea să descifreze structurile dumneavoastră de date XML pornind de la zero și fără a avea acces la DTD.

Într-un context modern, aceasta înseamnă că va trebui să țineți cont de numeroase dispozitive de afișare cu niveluri de abilități diferite. Aceste dispozitive pot fi:

- Afișaje numai de text sau auditive, cu o tastatură simplă.
- Web-TV (dispozitiv care furnizează pagini Web prin semnal TV) cu un dispozitiv de indicare.
- Terminale grafice de rezoluție înaltă cu tastatură și tabletă de digitizare și altele.
- Dispozitive Internet, dispozitive create pentru a rezolva câteva necesități specializate.
- Afișaje numai pentru text care cuprind atât terminale pentru persoane cu handicap de vedere, cât și dispozitive PDA (Personal Digital Assistant – asistent digital personal) sau chiar telefoane mobile.

Acest domeniu larg de dispozitive de ieșire potențiale presupune fie că va trebui să furnizați documentele dinamic, determinând în prealabil capacitățile browserului, fie să furnizați documentele la cel mai mic nivel, folosind tehnologii intermediare. Printre aceste tehnologii este XHTML care are și abilitatea de a include conținut XML sau de alt tip folosind informațiile de stil XSL pentru a permite modificarea dinamică a fluxului de date de intrare cu ajutorul unor dispozitive cu suport XML. Ca în cazul scripturilor de astăzi, browserele care nu vor suporta abilitățile avansate, le vor ignora.

Folosind tehnici similare, azi utilizatorii cu o vedere slabă sau chiar fără vedere se pot folosi de Internet la fel de bine ca persoanele cu vederea perfectă. Dacă nu puteți distinge între albastru și verde, de ce să nu știe asta și calculatorul dumneavoastră și să o compenseze într-un anumit fel? Cu ajutorul foilor de stil CSS și XSL, această compensare este banală atâta timp cât sensul de bază al fiecărui element de date este codat în pagină.

Sugestie de la LeeAnne.com Words to weave by

Accesibilitatea are un înțeles pentru toată lumea. Așa cum cei cu cărucioare pentru copii se folosesc de rampele pentru scaune pe rotile, asigurarea unui conținut accesibil poate ușura utilizarea sitului dumneavoastră în moduri pe care nici nu vi le imaginați.

Toate acestea sunt posibile. Tehnologia există. Mijloacele cuprind folosirea universală a XML în paginile Web. De exemplu, acum este destul de obișnuit ca autorii de documente Web să folosească indicii vizuale, cum ar fi culori, fonturi sau dimensiuni de litere pentru a sublinia înțelesul anumitor câmpuri de date. Din moment ce HTML nu vă permite să identificați ușor diferențele semnificative dintre tipurile de date, mecanismele de evidențiere tind să fie codate direct în pagină. Așa cum în calendare se obișnuiește ca sărbătorile și zilele de week-end să fie tipărite cu roșu, ceea ce nu are nici un sens dacă nu cunoașteți deja această convenție, autorii HTML au încercat să imite forțat convențiile din limba lor națională, în loc să permită ca înțelesul să se desprindă chiar din etichete, așa cum este posibil în XML.

Deci un autor HTML poate să codeze informația prea specific și fără să dea suficiente detalii:

```
<p>Click on the red text to select a link to an explanation <a href="URL"><font
color="red">here</font></a>.</p>
```

adică:

```
<p>Executati clic pe textul de culoare rosie pentru a selecta o legatura la o
explicatie <a href="URL"><font color="red">aici</font></a>.</p>
```

Pe de altă parte, un autor XML ar trebui să fie conștient că nu oricine are un mouse, deci „clic" nu este o descriere prea corectă pentru selectarea unei legături, iar folosirea cuvântului „here" (aici) pentru a descrie o legătură nu oferă prea multe informații despre ceea ce se află la celălalt capăt, atunci când este prezentată ca o listă izolată de legături, așa cum fac automat unele browsere auditive. Deci în XML ați putea scrie ceva de genul următor:

```
<paragraph>Select the following link to see a <a href="URL">further explanation</a>
of this subject.</paragraph>
```

adică:

```
<paragraph>Selectati urmatoarea legatura pentru <a href="URL">mai multe detalii</a>
despre acest subiect.</paragraph>
```

Observați că această versiune XML nu face nici o presupunere despre modul în care sunt evidențiate și afișate legăturile și oferă sintagma „further explanation" (mai multe detalii) pentru a descrie ceea ce se găsește la celălalt capăt al legăturii. Într-un exemplu mai lung, poate veți dori ca descrierile legăturilor să fie chiar mai formalizate, din moment ce ele pot fi prezentate separat ca o schiță, așa cum am menționat mai sus.

Accesibilitatea din punct de vedere uman

Browserele cu suport XML vă vor permite să definiți foi de stil XSL care iau în considerare propriile dumneavoastră abilități, pe cele ale dispozitivului de interfață actual, viteza conexiunii și multe alte variabile pentru a formata fiecare pagină așa cum este corect pentru dumneavoastră.

Există o sumedenie de motive excelente pentru a realiza pagini cât mai accesibile. În primul rând, lumea este plină de persoane cu abilități diferite. Chiar și limba este din ce în ce mai ușor de compensat, traducerile automate făcând navigarea prin situri în franceză sau rusă la fel de ușoară ca și citirea acestei cărți. Un vorbitor de limbă franceză ar putea folosi pagina dumneavoastră în engleză în același mod, interfața Web care intervine funcționând ca un fel de traducător universal care preia datele de intrare și le transformă într-un format pe care îl puteți utiliza.

Tratarea diferențelor de limbă în XML

V-ați gândit vreodată să publicați un document XHTML într-o altă limbă decât engleza? Dacă, de exemplu, aveți o mulțime de clienți în Germania, ar fi probabil în interesul dumneavoastră să aveți o versiune a propriului dumneavoastră sit, sau măcar a unei părți a sitului, tradusă în limba germană.

După cum poate știți, alfabetul german – deși este aproape identic cu cel englez – are în plus câteva litere, un accent, numit umlaut, și alte câteva caracteristici care îl deosebesc de alfabetul englez.

O rezolvare obișnuită când scrieți în germană este substituirea caracterelor specifice germane cu diagrafe formate din două caractere. Ess tzet, uneori denumit și scharfes ess (ß) este de obicei reprezentat prin „ss". O cu umlaut (ö) este de obicei tipărit „oe". Și așa mai departe. Acest tip de rezolvare este potrivit pentru o pagină care va avea o distribuție în lumea întreagă, deoarece caracterele germane speciale pot să nu fie disponibile pentru anumite browsere, în special pentru cele folosite de persoane cu handicap de vedere. Pe de altă parte, XML asigură facilități pentru a afișa orice caracter în aproape

orice limbă omenească, folosind o codificare Unicode, iar aceste codificări sunt adesea accesate prin mijlocirea entităților caracter mnemonice și nu direct prin valori numerice. Acest lucru înseamnă că valoarea unei anumite entități caracter mnemonice poate fi suprascrisă în submulțimea internă a DTD-ului, deci pot fi furnizate caracterele potrivite pentru fiecare browser prin suprascrierea valorilor prestabilite în submulțimea internă a DTD-ului, orice analizor XML trebuind să le recunoască, astfel:

```
Valoare prestabilită:  <!ENTITY szlig  CDATA "&#223;" -- latin small letter
                                       ➥sharp s = ess-zed, U+00DF ISOlat1 -->
Înlocuitor:            <!ENTITY szlig  CDATA "sz" -- latin small letter--
                                       ➥sharp s transliteration in ASCII  -->
```

Desigur, multe companii oferă și software pentru traduceri. Deși traducerea rezultată poate să nu fie la fel de expresivă și corectă ca una realizată de un traducător uman, este de obicei cel puțin inteligibilă. Dacă imaginea firmei este foarte importantă, trebuie să vă gândiți să angajați un traducător care să verifice rezultatele și să remedieze porțiunile dificile.

Puteți afla mai multe informații despre folosirea diferitelor limbi fie căutând în Help în Internet Explorer sau Netscape Navigator, fie pe siturile lor Web. Situl Opera Browser are o listă cuprinzătoare de versiuni pentru limbi europene la `http://www.opera.com`. Alis Technologies oferă de asemenea capacități multilingve în browserul lor Tango, printre care arabă și chineză: `http://www.alis.com`. W3C la `http://www.w3.org/WAI3` lucrează la standardele de limbă și accesibilitate pentru Internet. Dacă aceste aspecte sunt în general o parte importantă a sitului dumneavoastră Web, atunci merită să studiați informațiile și resursele lor la adresele `http://www.w3.org/WAI/` și `http://www.w3.org/International`.

Pentru a include un citat în germană într-un document XML de exemplu, următorul fragment va da câteva indicii agentului utilizator pentru a trata elegant un document în mai multe limbi:

```
<paragraph xml:lang="en" lang="en">Goethe once said in
<citation>Rechenschaft</citation>, <quote xml:lang="de" lang="de">Nur die Lumpen
sind bescheiden,/ Brave freuen sich der Tat.</quote></paragraph>
```

Deși probabil nu vor avea nici un efect vizibil, atributele `lang` de mai sus spun unui browser auditiv că textul citat se supune unor reguli de pronunție diferite față de regulile de pronunție în engleză, care guvernează restul paragrafului, iar celorlalte browsere le spun că există un citat inclus.

O foaie de stil asociată cu un document XML poate să țină cont de faptul că citatele respectă reguli diferite și sunt marcate cu caractere diferite în limbi diferite, astfel încât pentru a enumera cele patru stiluri europene de bază ați putea scrie:

```
:lang(en) > quote {quotes: '&lsquo;' '&rsquo;' '&ldquo;' '&rdquo;'}
:lang(en-US) > quote {quotes: '&ldquo;' '&rdquo;' '&lsquo;' '&rsquo;'}
:lang(fr) > quote { quotes: '&laquo; ' ' &raquo' }
:lang(de) > quote { quotes: '&raquo;' '&laquo;' '&rsaquo;' '&lsaquo;' }
```

Fragmentul de cod CSS de mai sus precizează succint că citatele în engleză folosesc un apostrof pentru citate de sine stătătoare și ghilimele pentru citatele incluse, în timp ce în engleza americană este exact invers. Dacă textul este în franceză, pentru a marca citatul se folosesc ghilimele unghiulare îndreptate în afară cu un spațiu inclus la fiecare capăt pe când în germană pentru a marca citatul se folosesc două niveluri de ghilimele unghiulare îndreptate în interior și fără un spațiu inclus. Deoarece limba codului din textul de bază

al fragmentului nu a fost specificată ca fiind engleza americană, se folosesc apostrofurile prestabilite din engleză, astfel:

```
Goethe once said in Rechenschaft, 'Nur die Lumpen sind bescheiden, Brave freuen
sich der Tat.'
```

Accesibilitatea din punct de vedere economic

Dată fiind evoluţia modurilor în care oamenii folosesc Internetul, de acum nu mai puteţi presupune nimic despre mediul în care vor fi rulate paginile dumneavoastră. Un utilizator poate să vă „citească" pagina deplasându-se cu automobilul pe şosea şi folosind un browser auditiv, la un colţ de stradă cu un PDA, printr-un telefon mobil, de la un frigider inteligent sau de la un set de bucătărie, sau dintr-un cărucior pe rotile în Braille. Figura 25.1 vă prezintă un telefon mobil Neopoint care accesează un e-mail; acelaşi telefon poate să răsfoiască pagini Web simple. Va fi necesară o pregătire specială din partea dumneavoastră, prin care să vă asiguraţi că pagina dumneavoastră va supravieţui dacă este afişată pe un mic ecran cu 11 linii.

După statisticile publicate de U.S. Bureau of the Census (Biroul de recensământ al Statelor Unite), la `http://www.census.gov`, aproape o cincime din populaţia SUA prezintă un anumit handicap, iar această proporţie este de aşteptat să crească la un sfert din total sau mai mult, având în vedere îmbătrânirea populaţiei. De fapt, dacă luaţi în considerare procesul normal de îmbătrânire, majoritatea persoanelor vor avea în cele din urmă un handicap fizic sau mintal de un anumit tip, din moment ce mai mult de jumătate dintre persoanele de peste 65 de ani prezintă un anumit nivel de handicap, iar o treime sunt cu handicapuri grave. Procente similare afectează majoritatea ţărilor din lume, iar multe naţiuni în curs de dezvoltare au procente mult mai mari de persoane cu handicap.

Sugestie de la LeeAnne.com Words to weave by

Îmbătrânind, oamenii au de obicei mai mult timp liber pentru a învăţa să folosească Internetul. Din cauza disponibilităţii informaţiilor şi serviciilor on-line, este posibil acum ca persoanele cu o mobilitate scăzută să câştige un acces independent la o comunitate mult mai largă decât în trecut. Deci proporţia persoanelor în vârstă care utilizează Internetul creşte într-o rată mai mare decât aţi putea deduce din cifre.

Asta nu înseamnă că oamenii nu mai cheltuiesc bani. Este chiar invers. Înaintând în vârstă, oamenii devin *mai* dependenţi de servicii, inclusiv de serviciile on-line, deoarece capacităţile lor se schimbă. Dacă odată îşi tundeau peluza singuri, acum trebuie să angajeze un serviciu de amenajare a curţii. Dacă înainte mergeau la cumpărături prin traficul aglomerat, acum fac comenzi. Aceasta este o piaţă care aşteaptă să fie fructificată din mai multe puncte de vedere. Marketingul focalizat actual este primitiv faţă de ceea ce va fi posibil în viitor.

Companiile trebuie să gândească rapid pentru a ţine pasul cu noile moduri în care pot să îşi servească baza de clienţi în continuă schimbare şi să şi câştige bani în acest timp. Având în vedere că informaţiile reale devin din ce în ce mai accesibile, fidelitatea faţă de o marcă s-ar putea să fie definitiv înfrântă. Atunci când agentul dumneavoastră inteligent poate să compare cu uşurinţă pe Web preţurile, înregistrările serviciilor, satisfacţia clienţilor, de ce să mai cumpăraţi de la cei de la care aţi cumpărat ultima dată? Este mai mult decât probabil ca oamenii să devină fideli faţă de o sursă sigură de informaţii, faţă de birouri de încredere care să ofere pentru o mică taxă informaţii confidenţiale despre sursele cele mai bune.

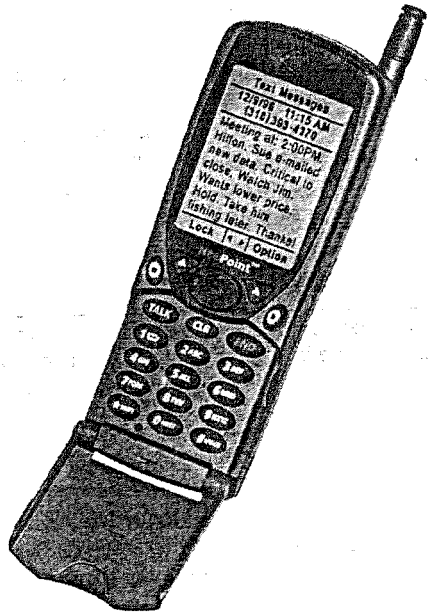


FIGURA 25.1
Iată un telefon mobil Neopoint care accesează Internetul pentru a citi un e-mail.

Strategiile de afaceri care se adresează acestor probleme pot să pornească de la asigurarea disponibilităţii şi accesibilităţii informaţiilor în toate tipurile de medii posibile, pentru a asigura o acoperire cât mai largă. De exemplu, puteţi furniza comparaţii de preţ şi de înregistrări de servicii prin legături pentru orice produs din ofertă, inclusiv datele competitorilor; şi să vă străduiţi să oferiţi un serviciu pe situl dumneavoastră Web care să facă clientul să revină. Nu vă mai puteţi baza numai pe afişe publicitare pompoase care să vă laude produsul. Trebuie să vă fundamentaţi afirmaţiile cu date şi cifre, aşa cum aţi face dacă aţi răspunde la o cerere din partea unei mari corporaţii.

Accesibilitatea din punct de vedere legal

În Statele Unite, Legea pentru americanii cu handicap (Americans with Disabilities Act - ADA) impune câteva cerinţe firmelor, agenţiilor guvernamentale şi instituţiilor de învăţământ pentru ca acestea să ofere un acces echivalent persoanelor cu handicap, atunci când acest acces poate fi disponibil cu promptitudine. Dacă există o tehnologie care face disponibil un acces echivalent şi nu o folosiţi, chiar dacă nu aveţi cunoştinţă despre aceasta, dar folosiţi în schimb o altă tehnologie, vi se poate cere să asiguraţi accesul chiar dacă noua dumneavoastră tehnologie, atât de grozavă, face asigurarea accesului echivalent dificilă şi costisitoare. Ca proiectant Web, este datoria dumneavoastră să vă cunoaşteţi instrumentele meseriei, iar recomandarea sau utilizarea unor tehnologii nepotrivite poate fi considerată o dovadă de incompetenţă profesională.

forma unui text. Cele mai importante mecanisme HTML sunt atributele `alt` și `longdesc`, dar poate fi utilizat și atributul `title`. În documentele XML generale aceste mecanisme nu există în mod prestabilit, dar pot fi asigurate cu ușurință prin DTD.

Sugestie de la LeeAnne.com Words to weave by

Deoarece există deja o convenție recunoscută în comunitățile HTML și XHTML, este logic, din punctul de vedere al lizibilității și ușurinței în utilizare, ca atributele HTML să fie copiate în instanțierile XML ale notațiilor, în afară de cazul unor considerații de suprascriere.

Notațiile folosite ca hărți-imagine trebuie să fie însoțite întotdeauna de o alternativă de navigare numai cu text.

Toate notațiile trebuie să conțină atribute cu un text alternativ succint (`alt="text"`). În XML nu există atribute predefinite pe care să vă puteți baza, deci un atribut `alt` ar trebui să fie declarat explicit la fiecare instanțiere a notației. Utilizați o descriere scurtă a imaginii în sine. Este de obicei o idee bună să folosiți un format standard pentru aceste descrieri, pentru a le deosebi de textul continuu. O sintaxă obișnuită este încadrarea textului în paranteze drepte, ca în

```
<img alt="[descrierea imaginii]" …>
```

sau adăugarea unei etichete la început pentru ca lucrurile să fie și mai clare:

```
<img alt="[IMAGINE: descrierea imaginii]" …>
```

Majusculele vor fi rostite la un nivel diferit față de textul continuu în multe browsere auditive, dând astfel o indicație clară că acest text este într-un anumit fel special.

O descriere mai completă a imaginii poate fi asigurată prin atributul `longdesc="url"` sau printr-o „legătură de descriere" dedicată. Deoarece XML nu asigură atributul `longdesc` sau notațiile, trebuie să declarați un atribut explicit `longdesc` la fiecare instanțiere a notației, astfel încât această informație să fie furnizată utilizatorului dacă este nevoie. Dacă doriți să evitați `longdesc`, puteți folosi o legătură. Plasați un „D" adiacent la imagine și legați „D"-ul la o pagină separată care conține o descriere detaliată a imaginii. Nici unul dintre browserele actuale importante nu suportă atributul `longdesc`, dar așa se intenționează să fie folosit. În XHTML, puteți furniza imaginea și „D"-ul asociat astfel:

```
<img src="URIimagine" alt="[IMAGINE: descrierea imaginii]" />
<a href="URItextulDescrierii">D</a>
```

Astfel asigurați atât o etichetă alternativă, utilă browserelor numai pentru text, cât și o descriere mai completă, care poate fi simplă, spre exemplu un paragraf care descrie o fotografie, sau mai complicată, cum ar fi un format alternativ pentru un tabel grafic.

Utilizarea accesibilă a legăturilor

Legăturile, fie că sunt implementate prin mecanismele de legare ale XML, fie prin elementul de hiperlegătură XHTML `anchor`, trebuie să conțină suficient text pentru a clarifica unde va ajunge utilizatorul dacă urmează legătura. „Click here" dă extraordinar de puține informații dacă este extras din context, ceea ce se întâmplă adesea în cazul multor agenți utilizatori auditivi, care furnizează o hartă de navigare rapidă a paginii. În JAWS, de exemplu, întreaga listă de legături este prezentată la primul acces la pagină, înainte de citirea textului efectiv al paginii. Astfel, navigarea este mai rapidă și veți trece mai rapid de la o pagină la alta. Dacă sunt folosite legături grafice, acestea trebuie să fie însoțite sau completate de legături text pentru cei care nu pot interpreta imaginile.

Un alt considerent este că trebuie să faceți legăturile destul de mari, astfel încât utilizatorii să manevreze cu ușurință un dispozitiv de indicare pentru a le selecta. Utilizatorii cu handicap motor sau de vedere au uneori dificultăți în observarea sau selectarea unor ținte de mici dimensiuni. Asigurarea accesului de la tastatură pentru elementele `link` este de asemenea bine gândită, deși nu toate browserele implementează scurtături de la tastatură prin atributul `accesskey`. Browserele care depind de Dynamic HTML pentru a gestiona accesul prin tastatură nu îl recunosc pentru utilizatorii care nu au un suport DHTML, deci par să nu își atingă ținta.

Utilizarea accesibilă a cadrelor

Multă lume are dificultăți cu cadrele, fie deoarece chiar cadrele îi încurcă, fie deoarece software-ul pe care îl folosesc nu le suportă. Dacă utilizați cadre în XHTML, trebuie să asigurați întotdeauna și o alternativă `NOFRAMES`. Utilizați `NOFRAMES` corect – mesaje îngâmfate de tipul „Modernizați-vă browserul" nu ajută cu nimic. `NOFRAMES` trebuie să conțină fie chiar conținutul paginii de intrare, fie legături la versiunile fără cadre ale paginilor dumneavoastră.

Utilizarea cu responsabilitate a cadrelor

Cadrele nu există independent în XML, dar pot fi implementate cu ușurință, dacă doriți. XHTML asigură un DTD pentru cadre, iar SMIL folosește un echivalent al acestora aproape identic. Totuși, cadrele sunt dificil de utilizat în multe situații. Unele browsere mai vechi nu le suportă, browserele auditive nu pot reda cadre, iar afișajele miniaturizate nu au destul spațiu pentru a le da un înțeles, chiar dacă ar putea. Adaptarea la cerințele tuturor utilizatorilor este la latitudinea creatorului paginii, care trebuie să se gândească bine atunci când creează o pagină cu cadre. Următoarele instrucțiuni vă dau câteva sugestii și sunt descrise în termenii cadrelor XHTML, dar se pot aplica și regiunilor de machetare din SMIL:

- Asigurați și un acces alternativ. Eliminați cadrele din sit pentru browserele care nu suportă cadrele. Copiați toate informațiile pertinente din fiecare cadru și adăugați-le la secțiunea `<NOFRAMES>` a paginii corespunzătoare. Etichetele `<FRAME>` și `<FRAMESET>` sunt ignorate dacă situl este vizualizat de un browser care nu suportă cadre, deci aceste elemente vor dispărea. Însă dacă sunt vizualizate de un browser care include suport pentru cadre, eticheta `<NOFRAMES>` va spune interpretorului HTML să ignore tot ce intră în domeniul său dar să afișeze ceea ce se află la locațiile `FRAMESET` și `FRAME`. De aceea trebuie să vă amintiți să folosiți mai întâi etichetele `<FRAMESET>` și `<FRAME>`, iar după aceea poate să urmeze alternativa HTML `NOFRAMES`. După ce ați terminat, veți avea o pagină care cuprinde atât elementele de navigare cât și toate informațiile.
- Salvați noul document. După ce informațiile din diferitele pagini cu cadre ale sitului au fost adăugate în pagina principală în ordinea lor logică, veți salva fișierul ca noua pagină prestabilită.
- Încărcați noul document. După ce ați realizat și ați salvat versiunea nouă a paginii cu cadre care încorporează și o alternativă fără cadre, trebuie să o faceți disponibilă sau – dacă operația este una de îmbunătățire – să înlocuiți versiunea mai veche care este deja disponibilă pe Net.
- Dacă situl este bine realizat, este posibil să trebuiască să modificați doar una sau două pagini pentru a vă asigura că întregul sit funcționează corect atât în browserele cu

cadre, cât și în cele fără cadre. Rețineți că pentru a realiza acest lucru trebuie să izolați pe cât posibil informațiile despre cadre.

Cadrele și SMIL au caracteristici comune prin faptul că încorporează informații de machetare chiar în document, deși SMIL refolosește o mare parte din mecanismul de machetare al CSS2. În ambele cazuri, nu este foarte ușor pentru utilizatori să modifice concepția impusă de autor pentru a se potrivi cu necesitățile lor. În concluzie, depinde de autor dacă permite vederi alternative în mod explicit sau forțează utilizatorul să-și adapteze mediul, sau chiar propria persoană, la pagină.

Folosirea JavaScript ca motor de redare

JavaScript este adesea folosit pentru a modifica sau a crea un mod de afișare a unui document și adesea este o problemă, deoarece majoritatea software-ului pentru utilizatorii Internet cu vedere slabă și pentru orbi nu poate să înțeleagă și să execute JavaScript. Ori de câte ori oferiți JavaScript, asigurați-vă că sunt disponibile și versiuni fără script-uri ale paginilor sau că documentul nu este modificat prin operațiile pe care le efectuați asupra lui. Deci un element decorativ JavaScript poate fi ignorat, în timp ce un mecanism de navigare JavaScript trebuie să aibă o alternativă numai cu text.

Aceleași avertismente se aplică și în cazul limbajelor Java, Perl, VBScript, Python, Tcl și al celorlalte limbaje de script. Dacă acestea constituie o parte din întregul mediului de redare ce ține de client, asigurați alternative. Dacă sunt utilizate pe partea de server, asigurați-vă că acțiunile lor țin cont de toți utilizatorii, fie prin determinarea browserului, fie programând pentru versiunea cea mai modestă.

Evitați fișierele în format PDF (Portable Document Format – format de document portabil)

În general, evitați cu orice preț utilizarea sau legarea la fișiere PDF. Acestea creează aproape întotdeauna probleme utilizatorilor cu vedere slabă sau orbi, necesitând jonglerii care consumă timp pentru ca textul să fie lizibil, deoarece acesta nu poate fi citit de un software pentru citirea ecranului fără o conversie prealabilă. Autorul poate să stabilească în fișier un indicator (furnizat de Adobe) care împiedică orice formă de redare accesibilă a documentului și, de multe ori, autorii folosesc această facilitate. Ar trebui să se numească „Puneți un bolovan în fața orbului". De fapt, Adobe încurajează utilizarea indicatorului pentru a împiedica utilizatorii să copieze ușor porțiuni de text, însă, în același timp, se blochează decodarea într-un format mai accesibil.

Deci foarte puține documente PDF pot fi citite de utilizatorii Internet, iar acestea numai folosind tehnologii de acces ajutate de Access Adobe. Access Adobe traduce PDF în ASCII sau HTML, făcând fișierul accesibil chiar dacă nu puteți accesa PDF în mod normal. Dacă oferiți fișiere PDF în situl dumneavoastră, asigurați-vă că există o legătură la `http://access.adobe.com` și că indicatorii sunt setați corect, astfel încât utilizatorii să poată extrage textul. Totuși, marea majoritate a fișierelor Adobe PDF disponibile pe Web sunt intenționat lăsate opace față de această facilitate, deci în viața reală este aproape inutilă.

Dacă mai adăugăm la toate acestea faptul că majoritatea utilizatorilor, nu numai cei cu handicap de vedere, consideră fișierele PDF iritante și lente la descărcare, este într-adevăr greu de înțeles de ce și-ar expune cineva situl atâtor critici justificate, pentru beneficii de o valoare atât de îndoielnică.

Evitați conținutul cu plug-in-uri multimedia

Shockwave, Flash, applet-uri Java, Dynamic HTML și alte plug-in-uri multimedia sunt în marea lor majoritate fără nici o valoare din punct de vedere al accesibilității. Fie asigurați un conținut alternativ, fie, mai bine, reconsiderați necesitatea lor. Oricum ele tind să fie greu de încărcat, și adesea nu contribuie chiar atât de mult la conținutul de informații al paginii.

Nu există o cale ușoară pentru a asigura un conținut accesibil folosind multe din aceste tehnologii. Singurul mod sigur pentru a furniza conținut multimedia este prin SMIL, dar și în acest caz, numai dacă includeți în document chiar de la început formate alternative. Puține situri par să se complice cu așa ceva, deci încurajarea folosirii lor este iresponsabilă.

În plus, deși acest lucru nu are un impact direct asupra accesibilității, utilizarea Java, JavaScript și a altor tehnologii în fișierele antet și a multor tehnici Dynamic HTML distruge utilitatea motoarelor de căutare din Internet, deoarece acestea de obicei nu știu ce să facă cu mesajele de eroare sau conținutul unor astfel de fișiere când sunt accesate de un spider (păianjen). Deși ați putea avea un conținut foarte frumos, dacă descrierea paginii dumneavoastră într-un răspuns la o căutare spune `Error: JavaScript incompatibility Please enable JavaScript in your Browser preferences, or download the latest version of Netscape or Internet Explorer ...` (`Eroare: Incompatibilitate JavaScript Activati JavaScript in preferintele browserului dumneavoastra sau descarcati ultima versiune Netscape sau Internet Explorer ...`) în locul unei descrieri clare a conținutului, ați pierdut o bună parte a valorii listingului din motorul de căutare.

Utilizarea Dynamic HTML pentru a implementa casete de text care sar, efecte de umbrire pentru ferestre, afișaje cu defilare, sau reîncărcarea automată a textului la intervale scurte este greu de înțeles dacă aveți vederea slabă. Nici browserele auditive nu se descurcă mai bine. Deci, dacă puteți, evitați folosirea acestor elemente.

Tabele

Utilizați cu multă precauție tabelele în documentele XML. O parte din software-ul care asigură acces la calculator și la Internet pentru utilizatorii cu handicap de vedere nu poate să citească tabelele în mod eficient, și adesea transformă tabelele dumneavoastră elegante într-o harababură. Dacă folosiți tabele pentru a formata textul continuu al sitului dumneavoastră, ar trebui să vă gândiți să furnizați și un conținut alternativ pentru cei care folosesc software pentru citirea ecranului. În XHTML și HTML, tabelele au fost dotate cu câteva atribute care ușurează navigarea și înțelegerea pentru persoanele care folosesc browsere auditive sau alte browsere care nu redau grafic documentul:

- Atributul `summary` al elementului `table` permite autorului să furnizeze o descriere scurtă a scopului tabelului și a datelor prezentate.
- Atributul `title`, existând pentru aproape orice element, permite ca tabelul să poarte un titlu iar fiecare celulă să fie identificată fără ambiguități.
- Atributele `accesskey` și `tabindex` permit navigarea prin tabele fără utilizarea unui mouse sau a altui dispozitiv de indicare.
- Atributul `abbr` asigură o referință prescurtată care poate fi înlocuită cu conținutul unei celule din capul de tabel, pentru a simplifica și scurta redarea auditivă.
- Atributul `axis` permite identificarea celulelor și în alt mod, pe lângă structura de bază a capului de tabel și permite o adnotare de categorie care descrie funcția exactă a celulei respective în tabelul care o conține.

- Atributele `headers` și `id` permit asocierea unei celule de date individuale cu oricâte celule de informații din capul de tabel identificate prin atributele `id`.
- Atributul `scope` permite unei celule din capul de tabel să definească domeniul de celule de date la care se referă, prin `col`, `row`, `colgroup` sau `rowgroup`.

Dacă folosiți tabele în documente generale XML, ar trebui să includeți în DTD atribute echivalente cu cele XHTML listate mai sus, probabil chiar cu același nume, pentru claritate și standardizare, și să le folosiți în întregul tabel. Listingul 25.1 vă arată cum ar putea fi folosite în practică aceste atribute.

Listingul 25.1 Un tabel XHTML care vă prezintă utilizarea atributelor pentru accesibilitate

```
<table summary="Table of rare books showing title, author,
 publication date, accession date, and location">
<caption>Rare Books in our Collections</caption>
<thead>
 <tr>
  <th id="title">Title</th>
   <th id=author">Author</th>
    <th id="PubDate">Publication Date</th>
     <th id="AccDate">Accession Date</th>
      <th id="CatNum">Catalog Number</th>
       <th id="Location">Location</th></tr>
</thead>
<tbody>
 <tr>
  <td id="Rerum" headers="Title">De rerum naturis</td>
   <td headers="Rerum Author">Rabanus Maurus</td>
    <td headers="Rerum PubDate">c. 1466 C.E.</td>
     <td headers="Rerum AccDate">1947 C.E.</td>
      <td headers="Rerum CatNum">RB1078</td>
       <td headers="Rerum Location">Locked Cabinet</td></tr>
 <tr>
  <td id="EtymOrig" headers="Title">Etymologiarum siue Originum</td>
   <td headers="EtymOrig Author">Isidorus Hispalensis</td>
    <td headers="EtymOrig PubDate">1186 C.E.</td>
     <td headers="EtymOrig AccDate">1967 C.E.</td>
      <td headers="EtymOrig CatNum">RB6721</td>
       <td headers="EtymOrig Location">Rare Book Room</td></tr>
<tr>
  <td id="PropRerum" headers="Title">Liber de proprietatibus rerum</td>
   <td headers="PropRerum Author">Bartholomaeus Anglicus</td>
    <td headers="PropRerum PubDate">1372 C.E.</td>
     <td headers="PropRerum AccDate">1887 C.E.</td>
      <td headers="PropRerum CatNum">RB008</td>
       <td headers="PropRerum Location">Rare Book Room</td></tr>
</tbody>
</table>
```

Deși este un singur nivel de clasificare, un agent auditiv poate să identifice fiecare celulă prin locația sa completă din matricea tabelului, pentru a evita confuziile. Deci un browser auditiv căruia i se cere să redea conținutul din a patra celulă pentru prima carte, ar putea spune „De rerum naturis, Accession Date, 1947 C.E." Pentru un tabel mai mare este bine să furnizați prescurtări pentru informațiile din capul de tabel, pentru a scurta timpul de redare într-un browser auditiv. Următorul fragment vă arată ce gen de informații puteți adăuga utilizând un atribut `abbr`:


```
<thead>
 <tr>
  <th id="title">Title</th>
   <th id=author">Author</th>
    <th id="PubDate" abbr="Publish">Publication Date</th>
     <th id="AccDate" abbr="Accession">Accession Date</th>
      <th id="CatNum" abbr="Catalog">Catalog Number</th>
       <th id="Location">Location</th></tr>
</thead>
<tbody>
 <tr>
  <td id="Rerum" headers="Title" abbr="Rerum">De rerum naturis</td>
   <td headers="Rerum Author">Rabanus Maurus</td>
    <td headers="Rerum PubDate">c. 1466 C.E.</td>
     <td headers="Rerum AccDate">1947 C.E.</td>
      <td headers="Rerum CatNum">RB1078</td>
       <td headers="Rerum Location">Locked Cabinet</td></tr>
```

În acest caz, un agent auditiv poate identifica aceeași locație ca „Rerum, Accession, 1947 C.E." Alte porțiuni din date ar putea fi identificate printr-un atribut `axis`.

Să facem față tehnicilor Braille

O dată cu dezvoltarea cititorilor de ecran și îmbunătățirea celorlalte soluții tehnice pentru orbi, tot mai puține persoane sunt dispuse să se dedice orelor intense de practică necesare pentru a ajunge la fluență în Braille. Este păcat, deoarece fără abilitatea de a citi și scrie fără ajutorul unui calculator, tot mai multe persoane cu vederea slabă sau oarbe devin analfabeți funcțional. Totuși, Braille este sistemul universal cel mai rapid de citire, după ce îl deprindeți. Este cu mult mai rapid decât cititorii de ecran, chiar și când viteza citirii este accelerată la cote ininteligibile. Și este mult mai ușor și mai rapid să scanați o pagină în căutarea unei anumite informații în Braille decât este să scotociți prin pagină cu un cititor de ecran. Poate apariția unor terminale Braille mai ieftine va răsturna această tendință spre analfabetism printre orbi.

XML și standardele legate de acesta încorporează suportul pentru dispozitivele de ieșire în Braille, deși implementarea driverelor pentru aceste dispozitive depinde de agentul utilizator. CSS2 și XSL conțin mecanisme de stil particulare pentru crearea sau suportul dispozitivelor de ieșire în Braille. În CSS2, tipurile media `braille` și `embossed` împreună cu grupul media `tactile` se referă la afișajele Braille. `braille` se referă la afișajul continuu, în marea majoritate dispozitive de ieșire cu o singură linie în Braille, în timp ce `embossed` se referă la ieșiri Braille în relief tipărite pe hârtie Braille. Deși schița recomandării XSL este încă în lucru, principalul mecanism XSL pentru accesibilitate pare să fie în prezent proprietatea `source-document`, care indică înapoi la documentele sursă XML care au fost folosite ca intrări. Este de presupus că astfel se va permite unui agent utilizator care are suport Braille să realizeze modificările de stil proprii și transformările documentului

original, poate inclusiv traducerea în Braille tipul 2 și în alte tipuri, ca proces separat.

Braille nu este un sistem monolitic, deși se lucrează în prezent la unificarea numeroaselor coduri Braille care sunt în folosință la ora actuală. Deși poartă același nume, există de fapt numeroase sisteme Braille pentru limbi diferite și pentru scopuri diferite. Chiar în aceeași limbă există sisteme de caractere cu șase puncte pentru fiecare caracter, un nivel elementar numit tipul unu și sisteme de tipul doi sau mai mari care folosesc adesea prescurtări și expresii idiomatice care marchează diferența dintre limba la nivelul de abecedar și limba literară.

Dacă mai adăugați și codul științific și pentru calculator Nemeth, Braille tipul 3, sistemele speciale cu opt puncte care extind caracterele de bază cu simboluri care nu sunt alfabetice și simboluri de formatare, reguli de indentare, simboluri de deplasare și multe altele, aveți de-a face cu o complexitate enormă. Există chiar și sisteme Braille speciale pentru muzică, coduri pentru calculator și matematică, care fac traducerea unei pagini într-un limbaj Braille literar sau tehnic la fel de dificilă ca și traducerea într-o limbă străină. De obicei, acestea nu pot fi realizate cu o foaie de stil, deși această părere este destul de răspândită. În majoritatea cazurilor trebuie să alcătuiți un program, deoarece pentru implementarea multor reguli sunt necesare anticiparea și backtracking-ul.

Oricum, o dată cu introducerea MathML, CML și a altor limbaje specializate de marcare bazate pe XML, posibilitățile de traducere pe calculator a textelor tehnice par foarte promițătoare. Deși sunt mult mai obositoare la citit, sistemele caracter cu caracter pot face publicațiile tehnice accesibile pentru utilizatorii orbi sau cu handicap de vedere care, altfel, ar fi avut nevoie de ajutorul unui cititor uman.

Discutarea regulilor și convențiilor folosite în tipărirea în Braille nu face parte din subiectul acestei cărți. Dacă doriți să aflați mai multe despre Braille tipul doi sau despre transcrierile tehnice pentru situl dumneavoastră, consultați articolele apărute în numeroasele publicații editate de American Printing House for the Blind.

Utilizarea foilor de stil auditive și a browserelor audio

Anexa D, „Referințe CSS1, CSS2", vă prezintă proprietățile folosite în foile de stil auditive. Veți observa că există proprietăți care controlează înălțimea sunetului vocii, poziția sa în spațiu și intensitatea sa.

Puteți folosi aceste proprietăți pentru a face mai clare elementele documentului pentru persoanele care îl studiază fără ajutorul văzului. Astfel, titlurile pot avea un volum mai mare decât textul normal, în timp ce textul în litere foarte mici va avea un volum mai scăzut. O adnotare poate fi poziționată în spațiu în extrema dreaptă și mai jos decât ascultătorul. Coloanele unui tabel, care sunt dificil de înțeles fără nici un ajutor, pot fi așezate în spațiu ca și cum ar fi răspândite în fața dumneavoastră, prima coloană fiind în stânga iar celelalte coloane trecând prin tot câmpul auditiv spre dreapta. Personajele dintr-o piesă de teatru pot să aibă propria lor voce, ritm și stil al vorbirii, modele de intonație și așa mai departe, iar pozițiile lor pe scena virtuală pot fi urmărite stereofonic.

Iată cum ar putea arăta un fragment din foaia de stil XML pentru piesa Hamlet, prințul Danemarcei:

```
dialog[speaker~="hamlet"] {pitch: 120Hz; pitch-range: 45; }
```



```
dialog[speaker~="polonius"] {pitch: 100Hz; pitch-range: 35; }
dialog[speaker~="gertrude"] {pitch: 200Hz; pitch-range: 60; }
dialog[speaker~="ophelia"] {pitch: 220Hz; pitch-range: 70; }
dialog[speaker~="ghost"] {pitch: 80Hz; pitch-range: 20; }
```

Aici, Hamlet are o voce normală de bărbat, Polonius sună mai gros și mai greoi, iar fantoma ar fi un *basso profundo* cu un domeniu de înălțimi monoton. Gertrude și Ofelia pot fi deosebite prin faptul că vocea Gertrudei este mai feminină în timp ce a Ofeliei sună mai aproape de a unei fete. Piesa poate fi pusă în scenă auditiv, plasând personajele într-un spațiu tridimensional, așa cum se mișcă ele pe scena imaginară.

```
<dialog speaker="hamlet" style="azimuth: left; ">To be or not to be,</dialog>
<dialog speaker="hamlet" style="azimuth: center; ">that is the question</dialog>
<dialog speaker="hamlet" style="azimuth: right; ">Whether tis nobler …</dialog>
```

Pentru simplitate și claritate, elementele de stil care mișcă în spațiu monologul lui Hamlet sunt plasate in-line în porțiuni de dialog, deși ar fi mai bine dacă ar fi plasate într-o foaie de stil asociată. Din nefericire, asta ar însemna trecerea înainte și înapoi de la foia de stil la document.

Această discuție atinge numai câteva aspecte superficiale și nu toate posibilitățile. Majoritatea motoarelor de redare auditive dispun de modele de voci cu nume, să zicem Karen, George sau Frank, care pot suferi modificări destul de complexe. Din nefericire, vocile nu sunt unitare în toate motoarele, la fel ca și fonturile alfabetelor, deci în cele din urmă va trebui să se creeze un sistem de etichetare general care să identifice o ingenuă, un clovn, un doctor și alte stereotipuri din *commedia dell'arte*. Astfel va exista o paletă de voci demnă de încredere, cu care se va putea lucra. Până atunci, vocile trebuie să fie sintetizate în funcție de caracteristicile destul de brute permise în lista de proprietăți CSS2.

Listingul 25.2 vă prezintă un exemplu mai lung, tot din Hamlet, care folosește etichetele XML ale piesei, plasate cu bunăvoință în domeniul public de Jon Bosak, având în plus câteva proprietăți auditive pentru a îmbogăți experiența și folosind foaia de stil fragmentară din primul exemplu. Discursurile au fost împărțite în linii, astfel încât software-ul de redare să poată emite corect metrica versului alb shakespearian. Utilizarea in-line a codului de stil, care este descurajată, este folosită aici numai pentru a vedea ceea ce se întâmplă, pentru a evita să vă uitați înainte și înapoi de la foaia de stil la document.

Listingul 25.2 Scena fantomei din Hamlet redată în XML

```
<dialog speaker="hamlet" style="azimuth: left; ">
<line>Where wilt thou lead me? speak; I'll go no further.</line>
</dialog>
<dialog speaker="ghost" style="azimuth: center; elevation: -45deg ">
<line>Mark me.</line>
</dialog>
<dialog speaker="hamlet">
<line>I will.</line>
</dialog>
<dialog speaker="ghost">
<line>My hour is almost come,</line>
<line>When I to sulphurous and tormenting flames</line>
<line>Must render up myself.</line>
</dialog>
<dialog speaker="hamlet">
```

Listingul 25.2 (continuare)

```
<line>Alas, poor ghost!</line>
</dialog>
<dialog speaker="ghost">
<line>Pity me not, but lend thy serious hearing</line>
<line>To what I shall unfold.</line>
</dialog>
<dialog speaker="hamlet">
<line>Speak; I am bound to hear.</line>
</dialog>
<dialog speaker="ghost">
<line>So art thou to revenge, when thou shalt hear.</line>
</dialog>
<dialog speaker="hamlet">
<line>What?</line>
</dialog>
<dialog speaker="ghost">
<line>I am thy father's spirit,</line>
<line>Doom'd for a certain term to walk the night,</line>
<line>And for the day confined to fast in fires,</line>
<line>Till the foul crimes done in my days of nature</line>
<line>Are burnt and purged away. But that I am forbid</line>
<line>To tell the secrets of my prison-house,</line>
<line>I could a tale unfold whose lightest word</line>
<line>Would harrow up thy soul, freeze thy young blood,</line>
<line>Make thy two eyes, like stars, start from their spheres,</line>
<line>Thy knotted and combined locks to part</line>
<line>And each particular hair to stand on end,</line>
<line>Like quills upon the fretful porpentine:</line>
<line>But this eternal blazon must not be</line>
<line>To ears of flesh and blood. List, list, O, list!</line>
<line>If thou didst ever thy dear father love—</line>
</dialog>
<dialog speaker="hamlet" style="volume: loud; ">
<line>O God!</line>
</dialog>
<dialog speaker="ghost">
<line>Revenge his foul and most unnatural murder.</line>
</dialog>
<dialog speaker="hamlet" style="azimuth: center; volume:x-loud; ">
<line>Murder!</line>
</dialog>
<dialog speaker="ghost" style="azimuth: right; ">
<line>Murder most foul, as in the best it is;</line>
<line>But this most foul, strange and unnatural.</line>
</dialog>
<dialog speaker="hamlet" style="azimuth: far-right; volume: medium; ">
<line>Haste me to know't, that I, with wings as swift</line>
<line>As meditation or the thoughts of love,</line>
<line>May sweep to my revenge.</line>
</dialog>
```

În acest fragment de dialog, vocea fantomei este plasată în poziția tradițională, sub nivelul scenei, iar personajele se mișcă individual, Hamlet pășind dintr-o parte a scenei în alta în timp ce fantoma se îndepărtează; din nou punerea în scenă este tradițională. De asemenea, Hamlet ridică vocea când este tulburat.

Unul dintre programele de sinteză vocală existente este JAWS de la Henter-Joyce, la `http://www.hj.com/`, fiind modernizat recent pentru a asigura o accesibilitate Web mai bună, iar altul este pwWebSpeak de la Productivity Works, la `http://www.prodworks.com/`.

Sugestie de la LeeAnne.com Words to weave by

pwWebSpeak a primit aprecieri foarte bune în comunitatea celor cu handicap de vedere/orbi. Productivity Works dispune de produse pentru navigarea non-tradițională pe Web, inclusiv pwTelephoneSpeak, un browser creat pentru utilizarea prin telefon. Pentru mai multe informații, vizitați `http://www.prodworks.com/`.

BIBL. CENTR. UNIV. "CAROL I" IAȘI MATEMATICĂ

Editarea vocilor pentru redarea auditivă

Redarea fină a foilor de stil XML poate fi făcută prin modificarea parametrilor permiși chiar în CSS2 sau XSL, sau prin construcție, plecând de la o voce de bază care există deja în pachetul auditiv, dacă sunteți sigur că acea voce va fi disponibilă. Deci puteți selecta vocea lui Shelly, un personaj *JAWS*, cu o valoare feminină prestabilită, dacă Shelly nu există în motorul auditiv. Deoarece vocea lui Shelly poate să nu vi se pară perfectă pentru Ofelia, puteți să îi setați înălțimea normală a vorbirii ceva mai sus decât media iar nivelul de animație să fie foarte înalt, astfel încât intervalul de frecvențe pe care le folosește să fie destul de larg:

```
dialog[speaker="ophelia"] { voice-family: shelly, female; pitch: 230hz; pitch-
range: 80; }
```

Aproape toate pachetele de sinteză vocală vă permit să modificați parametrii vocilor. *JAWS*, de exemplu, conține o selecție de personaje: Reed, Shelly, Rocko, Bobby, Glen, Sandy, Grandma și Grandpa. Puteți seta ritmul și frecvența, dar alți parametri nu sunt disponibili, în afara unui nivel de frecvență general pentru majuscule tastate cu Shift sau Caps Lock. Shelly, Sandy și Grandma sunt voci feminine iar celelalte sunt masculine, fiind mai îngroșate. Puteți de asemenea să stabiliți reguli de pronunție, deci dacă citiți în spaniolă, pronunția va fi mai apropiată de cea a unui vorbitor nativ. Puteți alege chiar între engleza americană și cea britanică.

De la tastatură, puteți seta *JAWS* să gestioneze diferite evenimente cu voci diferite, de exemplu, mesajele de sistem vor suna altfel decât setarea globală folosită pentru citirea textului. Produsul este ușor de utilizat și în mare parte intuitiv. Pentru a citi o linie trebuie să mutați cursorul în orice punct al ei folosind tastele săgeată-sus și săgeată-jos. Pentru a rosti cuvinte, mutați cursorul cu tastele săgeată-stânga sau săgeată-dreapta.

Manipularea mediilor țintă multiple

În prezent există destul de puține mecanisme sigure care să asigure foi de stil variabile pentru diferite medii. Până când vor fi disponibile browsere XML care suportă aceste medii, va trebui să vă bazați pe medii codate hard, cum au fost exemplele foilor de stil auditive discutate mai sus, pe foile de stil ale utilizatorului și pe ingeniozitatea utilizatorilor în manipularea mediilor multiple.

În secțiunea următoare, veți studia pașii necesari în crearea efectivă a unui document accesibil.

Realizarea unui document XML accesibil

Pentru acest studiu de caz, veți crea o pagină care va fi accesibilă pentru mai multe tipuri de browsere, va da indicii de navigare bune pentru oricine, inclusiv pentru persoanele cu handicap de vedere, și va asigura un conținut alternativ și posibilități de navigare alternative chiar și pentru persoanele care dispun de browsere numai pentru text, cum este Lynx, care nu are suport pentru cadre. Veți folosi XHTML simplu, pentru o compatibilitate cât mai bună cu browserele existente, deși codul documentului poate fi creat pe loc, cu un șablon XML folosind XSLT. Pentru a verifica accesibilitatea sitului, veți folosi setul de validare Bobby, un instrument disponibil pe scară largă, proiectat pentru a identifica eventualele probleme pe care le pot avea utilizatorii cu vederea slabă sau orbi, atunci când vizitează situl.

Următorul sit Web este disponibil la `http://www.igc.apc.org/women/bookstores/feminist-frames.html` și este folosit ca și conținut pentru documentul XHTML pentru orice browser.

Librăria feministă din Africa

Listingul 25.3 reprezintă o pagină XHTML reală, proiectată pentru a fi accesibilă în cât mai multe browsere, deși folosește cadre, care sunt de obicei dificil de manipulat pentru multe browsere auditive. Realizează acest lucru folosind eticheta `<noframes>`, care identifică conținutul ce va fi redat de browserele care nu au suport pentru cadre.

Listingul 25.3 Textul de prezentare al librăriei

```
Feminist Bookstores Index
    SUB-SAHARAN AFRICA
    (Including Madagascar)
    Countries Indexed:
        Kenya
    Kenya
    Country Telephone Code 254
    Nairobi, Nairobi Province, Kenya
        Binti Legacy
        (Feminist Bookstore)
        Loita House
        Loita Street
        Nairobi
        Kenya
        +254.2.33.0854 FAX
        +254.2.33.0854
    Contact for Binti Legacy in the USA:
        Dan Hoffman
        597 San Luis Road
        Berkeley, CA  94707
        USA
        +1.510.527.7651 FAX
        +1.510.528.9900
    African Feminist Publishers

        (None Known)
```


Acest mic fragment este luat dintr-un exemplu real de pe Web, care include multe alte librării; l-am ales pe acesta deoarece era cea mai scurtă listă. Când am creat lista am folosit un format compatibil cu Lynx, foarte agresiv, care nu este foarte reușit, deoarece nu conține nici o imagine și nici o formatare. În același timp, nu am dorit să adaug prea multe zorzoane, deoarece scopul paginii este pur informativ. Deci am făcut un compromis. Însă am intrat într-o eră nouă a dezvoltării Web-ului și m-am gândit că situl are nevoie de un stil mai atractiv, deci trebuia să îl modernizez.

Pentru a actualiza formatul am folosit cadre, care nu sunt chiar atât de greu de manipulat corect, iar pentru a face prezentarea mai interesantă am creat în JavaScript o bară de navigare care evidențiază ceea ce este indicat de mouse. Părea destul de bine pentru o pagină simplă, dar punea deja câteva probleme care nu puteau fi evitate cu ușurință. Prima era ce se întâmplă cu browserele care nu suportă cadre.

Gestionarea cadrelor în browserele care nu suportă cadre

Această problemă este destul de simplă. Atunci când cei de la Netscape au creat etichetele pentru cadre, le-au proiectat astfel încât să fie compatibile cu versiunile mai vechi de browsere. Deoarece etichetele `<frame>` și `<frameset>` nu includ un conținut in-line, sunt ignorate de browserele care nu le înțeleg. În schimb, browserele care suportă cadre ignoră orice conținut in-line încadrat de o pereche de etichete `<noframes>`. Deci tot ce aveți de făcut este să includeți versiunea fără cadre a paginii într-un container `<noframes>`.

Cealaltă problemă era ce se poate face cu Microsoft Internet Explorer. Din mai multe motive, cei de la Microsoft nu au ținut pasul cu dezvoltarea JavaScript și nu au putut manipula prea bine caracteristicile mai avansate ale JavaScript în versiunile mai vechi ale browserului lor. Aș fi putut introduce script-uri mai complicate care să trateze totul în document dar ar fi devenit rapid un coșmar pentru întreținere. De câte ori introduceți un cod pentru a trata un bug, trebuie să vă hotărâți dacă urmăriți apariția bug-urilor, dar astfel ciclul dumneavoastră de dezvoltare devine dependent de toanele unei alte companii, sau dacă tratați un bug și pe cele care apar mai târziu le ignorați, o soluție tot atât de puțin satisfăcătoare.

Iată ce am ales să fac: deoarece browserul Microsoft crea probleme, am separat tratarea bug-ului Microsoft și am pus-o în pagini de navigare create doar pentru Microsoft. În acest fel, puteam să le urmăresc bug-urile sau să le ignor în deplină siguranță.

Deci am avut nevoie doar de un mic fragment JavaScript care să comute browserele Microsoft în versiunea lor de sistem de navigare cu cadre și, pentru a completa, să comute toate celelalte browsere în afara versiunilor Microsoft, dacă din anumite motive s-ar rătăci prin acestea. Iată fragmentul care detectează browserele Microsoft. Acesta funcționează astfel: caută în numele browserului cuvântul „Microsoft", iar dacă îl găsește comută la versiunea Microsoft:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html
 xmlns="http://www.w3.org/TR/xhtml1">
  <head>
    <script language="JavaScript">
      <!--
        var browser = navigator.appName;
```

```
        if (browser.indexOf ("Microsoft") != -1)
          location = "feminist-frames-msie.html";
      // -->
    </script>
...
```

Fragmentul corespunzător care comută la versiunea standard caută de asemenea cuvântul „Microsoft", dar comută numai dacă nu îl găsește. Sună complicat, dar funcționează pentru majoritatea browserelor. Acele browsere care nu suportă JavaScript vor ignora acest comutator și nu vor vedea nici o diferență între cele două versiuni, deoarece acestea diferă doar prin modul de funcționare al JavaScript:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html
 xmlns="http://www.w3.org/TR/xhtml1">
  <head>
    <script language="JavaScript">
      <!--
        var browser = navigator.appName;
        if (browser.indexOf ("Microsoft") == -1)
          location = "feminist-frames-msie.html";
      // -->
    </script>
```

Definirea aspectului paginii

Acum putem începe munca adevărată, definirea aspectului paginii. Mai întâi am codat pagina de bază africană. Încep întotdeauna cu cele mai simple lucruri pentru că știu că funcționează, iar apoi adaug etichete și alte funcționalități. Am ținut cont de regulile Bobby (`http://www.cast.org`) pe parcursul scrierii codului și l-am validat de mai multe ori în timp ce lucram la el, pentru a mă asigura că nu am uitat nimic.

Deoarece mulți utilizatori cu handicap de vedere vor găsi chiar pagina informativă, folosind sistemul meu alternativ de navigare, am dorit să fiu foarte sigură că este accesibilă. După ce am terminat și am trecut prin toate testele Bobby, am inserat fragmentul de cod pe care acestea le furnizează pentru a semnala conformitatea cu regulile lor și m-am felicitat. Prima sarcină era îndeplinită!

Notă

Nu pățiți nimic dacă vă reamintiți la ce sunt bune practicile pentru proiectare accesibilă. La `http://www.w3.org/WAI/` există o listă care cuprinde un sumar și câteva informații în plus. Lucrurile de bază sunt evidente dacă vă gândiți puțin, dar sunt și aspecte mai subtile. Pentru început, nu trebuie să folosiți mai mult de o legătură pe o linie; folosiți atribute alt descriptive pentru toate imaginile sau o descriere vidă explicită, un contrast bun între text și fundal, fundal fără modele sau texturi și mijloace alternative de acces la elementele de navigare vizuale. Mai sunt multe informații utile în acest sit, deci vizitați-l înainte de a petrece prea mult timp proiectând pagini. Multe trucuri favorite ale creatorilor sunt fie interzise, fie nu sunt recomandabile, deci merită să studiați aceste probleme înainte de a vă apuca de un proiect important. Pe de altă parte, uneori sunt necesare și elemente care produc probleme. În acest caz trebuie să asigurați pagini alternative.

După ce informația din pagină a fost introdusă, am început cu cadrele înconjurătoare, începând cu pagina de bază și trecând apoi la paginile de navigare, deci întregul proces a trecut din interior spre exterior, iar apoi din exterior spre interior.

Am decis ca bara de navigare să fie pe marginea din stânga, o alegere destul de comună, iar bara de navigare secundară, mai mică, am plasat-o pe marginea de jos. Ambele bare de navigare trăiesc în propriile lor cadre iar restul ecranului este dedicat informațiilor.

Cadrele și JavaScript le-am tratat deja, deci vă voi prezenta aici numai fragmente. Întregul sit poate fi văzut pe Web la `http://www.igc.apc.org/women/bookstores/feminist-frames.html` dacă doriți să explorați restul codului. O parte din fișierele grafice și din codul din această carte sunt disponibile și pe situl meu Web: `http://www.leeanne.com/practical`.

Listingul 25.4 vă prezintă prima parte a paginii de intrare pentru situl cu cadre.

Listingul 25.4 Pagina de intrare XHTML pentru lista de librării

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html
 xmlns="http://www.w3.org/1999/xhtml">

<frameset
   cols="100,*"
   frameborder="0"
   framespacing="0"
   border="0">
    <frame
     src="bookmenu.html"
     marginheight="0"
     marginwidth="0"
     noresize="noresize"
     name="MenuBar"
     noresize="noresize"
     scrolling="no" />
    <frameset
     rows="*,20"
     frameborder="0"
     framespacing="0"
     border="0">
      <frame
       src="bookmain.html"
       marginheight="0"
       marginwidth="0"
       noresize="noresize"
       name="MainWindow"
       noresize="noresize" />
      <frameset
       frameborder="0"
       framespacing="0"
       border="0">
        <frame
         src="booknavigate.html"
```

Listingul 25.4 (continuare)

```
        marginheight="0"
        marginwidth="0"
        noresize="noresize"
        name="Navbar2"
        scrolling="no" />
    </frameset>
  </frameset>
  <noframes>
    <body
     bgcolor="FFFFFF">
      <!--Begin Index-->

      <p>
        <br />
      </p>
      <blockquote>
        <a href="#Index"><img src="fembooks.gif" height="124"
         width="354" align="left"
         alt="Feminist Bookstore Index Menu" border="0" /></a>
      </blockquote>
      <h1 align="center">Feminist Bookstores Index</h1>
      <p>
        <br clear="clear" />
      </p>
      <p align="right">
         <a href="widehtml.html" target="_top"><img src="textonly.gif"
         height="25" width="80" align="right" border="0" hspace="10"
         vspace="0" alt="Text Only Version" /></a><a
         href="feminist-text.html" target="_top">Text Only Version</a>
      </p>
      <p align="right">
         <a href="feminist-frames.html" target="_top"><img
         src="frames.gif" height="25" width="80" align="right"
         border="0" hspace="10" vspace="0"
         alt="Frames Version" /></a><a href="feminist-noframes.html"
         target="_top">Frames Version</a>
      <a name="Top"></a>
      </p>
      <blockquote>
        <h2>Welcome! </h2>
      </blockquote>
      <blockquote>
        <p>
          You've reached this page from a browser without frames
          support. That's fine with me. All the information on
          this site is accessible to all browsers that I'm aware
          of, including text browsers like Lynx and as accessible
          for the vision-impaired as possible.
        </p>

...
```


Codul se poate opri aici, deşi pagina continuă, deoarece nu sunt prea multe de învăţat din datele efective conţinute în pagină.

Paginile de navigare pentru browserele care fie suportă corect JavaScript (fie îl ignoră complet) arată ca în Listingul 25.5.

Listingul 25.5 Pagina de navigare a sitului librăriei

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/transitional.dtd">
<html
 xmlns="http://www.w3.org/1999/xhtml">
  <head>
<script
 language="JavaScript"
 type="text/javascript">
<!--
var browser = navigator.appName;
if (browser.indexOf ("Microsoft") != -1)
       location = "feminist-frames-msie.html";
// -->
</script>
<script
 language="JavaScript"
 type="text/javascript">
<!--
var image1 = new Image(88,86);
var image2 = new Image(88,30);
var image3 = new Image(88,30);
var image4 = new Image(88,30);
var image5 = new Image(88,30);
var image6 = new Image(88,30);
var image7 = new Image(88,30);
var image8 = new Image(88,30);
var image9 = new Image(88,30);
var image10 = new Image(88,30);
function init() {
   image1.src = "menu-logohigh.gif";
   image2.src = "menu-africahigh.gif";
   image3.src = "menu-asiahigh.gif";
   image4.src = "menu-austrnzhigh.gif";
   image5.src = "menu-canadahigh.gif";
   image6.src = "menu-europehigh.gif";
   image7.src = "menu-mideasthigh.gif";
   image8.src = "menu-samericahigh.gif";
   image9.src = "menu-usahigh.gif";
   image10.src = "menu-linkshigh.gif";
   }
init();
function highlight(image) {
   document[image].src = image + "high.gif";
   }
function unhighlight(image) {
```

Listingul 25.5 (continuare)

```
    document[image].src = image + "low.gif";
    }
// -->
</script>
    <title></title>
  </head>
  <body bgcolor="#00ffff">
    <a href="bookmain.html" target="MainWindow"
       onmouseover="highlight('menu-logo'); self.status='Feminist \
       Bookstores Index Worldwide Home'; return true;"
       onmouseout="unhighlight('menu-logo'); self.status=''; return
       true;"><img name="menu-logo" src="menu-logolow.gif" hspace="0"
       vspace="0" width="88" height="86" border="0"
       alt="Feminist Bookstores Index Worldwide Home" /></a><a
       href="africa.html" target="MainWindow"
       onmouseover="highlight('menu-africa');
       self.status='Sub-Saharan Africa - Feminist Bookstores';
       return true;"
       onmouseout="unhighlight('menu-africa'); self.status='';
       return true;"><img name="menu-africa" src="menu-africalow.gif"
       hspace="0" vspace="0" width="88" height="30" border="0"
       alt="Africa - Feminist Bookstores" /></a><a
       ...
```

Nu vă arăt întreaga pagină de navigare, iar a doua bară din partea de jos a ecranului este aproape identică, deci dacă sunteți curios, puteți să le studiați pe situl Web unde puteți vedea în acțiune întreaga poveste.

La fiecare pas am testat fiecare pagină cu instrumentul de validare Bobby. Întregul sit trebuie să treacă testele pentru a putea afișa logo-ul, deci este important să testați toate paginile, nu numai pe cele de bază.

Rezultatul final este prezentat în Figura 25.2.

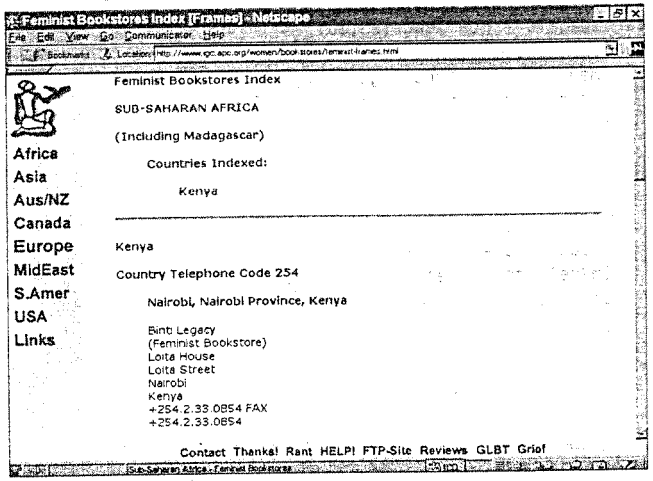


FIGURA 25.2
Situl cu cadre al librăriei feministe vă prezintă pagina africană.

Pe pagina de intrare vizitatorul poate opta între versiunile sitului fără cadre, numai text și normală. Figura 25.3 vă prezintă versiunea cu cadre a sitului, cu legături la versiunile fără cadre și numai text. În cea mai mare parte, toate aceste versiuni folosesc aceleași fișiere, sau sunt create prin modificarea automată a unui fișier de bază. Sunt doar câteva fișiere care tratează diferite versiuni iar aceste fișiere sunt modificate foarte rar.

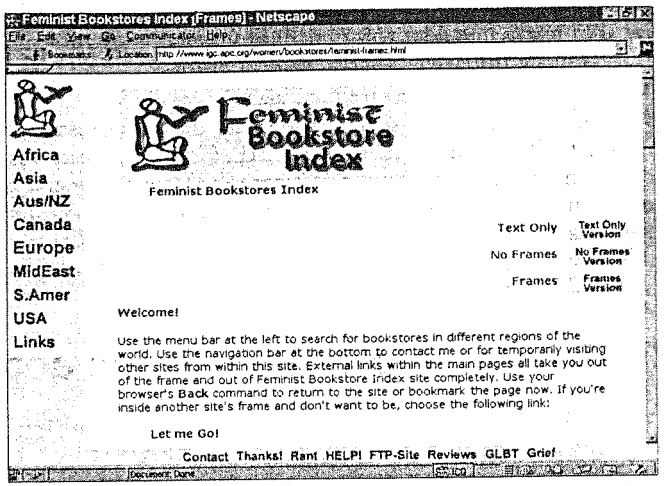


FIGURA 25.3
Situl cu cadre al librăriei feministe.

În final, am să vă prezint în Figura 25.4 o captură a ecranului cu logo-ul Bobby, ca să vedeți ce veți primi drept răsplată pentru munca dumneavoastră, în afara satisfacției că ați făcut ceea ce trebuia.

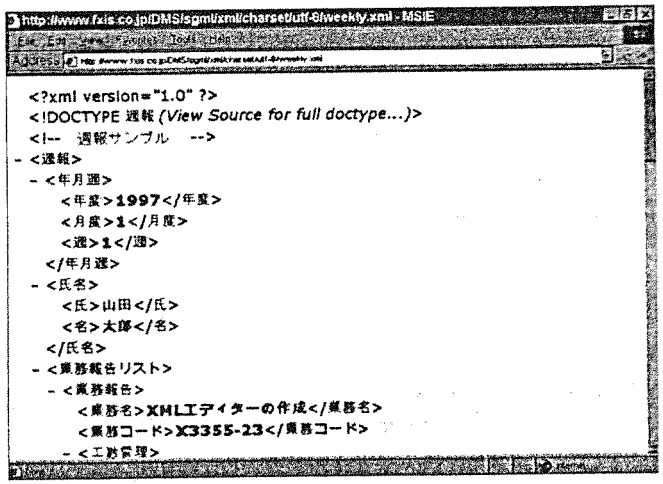


FIGURA 25.4
Logo-ul Bobby pe situl librăriei feministe.

Asta este! Accesibilitate într-o coajă de nucă. Deși exemplul prezentat este în XHTML, deoarece toate instrumentele disponibile pentru testarea automată a accesibilității se

bazează pe HTML/XHTML, aceleași tehnici pot fi folosite și pentru implementarea paginilor accesibile în orice vocabular XML.

Accesibilitatea este în același timp și greu și ușor de manipulat. Există resurse pe Web, cum ar fi situl Bobby, menționat în exemplul sitului librăriei, care automatizează procesul pentru paginile HTML și XHTML. În XML este mai problematic, dar întregul sit sau doar fragmentul cel mai reprezentativ poate fi convertit în HTML folosind XSLT iar apoi poate fi validat folosind Bobby sau alte instrumente listate pe situl W3C Web Accessibility Initiative la `http://www.w3.org/WAI/`.

Multe instrumente de creare încurajează accesibilitatea, iar câteva sunt chiar insistente. Dacă urmați sugestiile și indicațiile date de aceste instrumente nu veți greși aproape niciodată. Printre instrumentele HTML, HoTMetaL este poate cel mai de seamă pentru devotamentul cu care asigură atât accesibilitatea în general cât și accesibilitatea instrumentului în sine. Co-fondatorul lui SoftQuad Inc., Yuri Rubinsky, a cărui fundație Yuri Rubinsky Insight Foundation, la `http://128.100.250.10/yuri/`, este activă și în prezent în problemele persoanelor cu handicap, s-a asigurat că toate produsele companiei sale sunt accesibile pentru oricine.

Rezolvarea problemelor de accesibilitate

Rezolvarea plângerilor utilizatorilor cu handicap

În afară de cazul în care sunteți foarte grijuliu cu testarea, situl dumneavoastră ar putea atrage o plângere din partea unui utilizator cu handicap, care în mod justificat se simte nedreptățit și nu ezită să vă anunțe acest lucru.

Acordați-i atenție. Dacă este ceva ce nu înțelegeți, răspundeți printr-o întrebare politicoasă care să arate că sunteți dispus să învățați. Persoanele cu handicap au mult mai multă experiență decât dumneavoastră în tratarea handicapului lor. Adesea, sunt surse bogate de informații legate de rezolvarea ușoară și ieftină a problemelor pe care le au. De asemenea, puteți considera că v-au făcut o imensă favoare. Mulți dintre clienții dumneavoastră potențiali, poate sute, au trecut mai departe fără să se deranjeze să vă scrie observațiile lor. Sau poate nu au avut unde să-și trimită observațiile. Asigurați-vă că dispuneți de informații de contact, etichetate clar în pagina de intrare a sitului dumneavoastră.

Notă O pagină de intrare (splash page) este pagina de bază a unui sit, prima pagină pe care o vede cineva, și care stabilește identitatea și aspectul general al sitului. Desigur, știți deja că folosirea unei pagini de intrare cu redirectare, care afișează o pagină de prost gust, sau chiar mai rău, o serie de astfel de pagini de intrare, este o nemaipomenită risipă de timp și de resurse.

Deși orice sit ar trebui să asigure o identitate pentru e-mail `webmaster@{numesit}` pentru a simplifica viața persoanelor cu handicap care poate au nevoie de ajutor, este surprinzător cât de multe situri fac dificilă contactarea unui om. Unele folosesc formulare situate în altă pagină, care poate fi chiar și ea inaccesibilă, ceea ce este de-a dreptul scandalos.

Rezolvarea cererilor de conținut inaccesibil ale clientului

Clientul dumneavoastră decide că dorește în fiecare pagină elemente de navigare Shockwave care cântă și dansează, un fundal negru și un text gri închis, pentru că arată „artistic" și multe alte caracteristici despre care știți că vor crea dificultăți pentru utilizatorii cu handicap.

Mai întâi încercați să vă convingeți clientul. Arătați-i problemele potențiale care apar din cauza elementelor propuse și costul întreținerii unor alternative de navigare pentru fiecare pagină. Dacă nu reușiți, gândiți-vă să îl puneți să semneze că vă absolvă de orice responsabilitate și nu vă scrieți numele ca proiectant al paginii, rămânând într-un anonimat foarte plăcut. Aceste lucruri se răspândesc foarte repede, iar atunci când clientul va fi dat în judecată nu este bine să fiți prin preajmă, ca să fiți învinovățit.

Pentru mai multe informații despre afișarea conținutului pentru utilizatorii cu vedere slabă **vezi** „Tehnici de afișare pentru vedere slabă", la **pagina 588.**

Cooperarea în domeniu şi resurse

CAPITOLUL **26**

În acest capitol

Trecerea spre XML în aplicaţiile comerciale
DTD-uri de succes
Alte grupuri legate de XML din diverse domenii
Arhive DTD şi organisme de standardizare
Software legat de XML
Studii de caz

Trecerea spre XML în aplicaţiile comerciale

Perspectivele pentru acceptarea pe scară largă a DTD-urilor XML şi a schemelor XML ca limbaje comune de comunicare sunt şi foarte bune, şi foarte proaste, în funcţie de cine sunteţi ca utilizator potenţial al noilor standarde XML. DTD-urile cele mai reuşite sunt legate de canale furnizor/cumpărător în care schema XML este impusă de partea care este mai puternică din punct de vedere economic. În cazul guvernului SUA, acesta nu are prea multe dificultăţi în a impune CALS; guvernul a anunţat pur şi simplu furnizorii că, dacă în ofertele lor nu respectă acest standard, nu vor fi luaţi în considerare. La fel, aproape orice cumpărător important poate să impună furnizorilor săi formate de transfer de date. Iar furnizorii vor fi foarte mulţumiţi să accepte aceste formate, deoarece au de câştigat o grămadă de bani atunci când asigură servicii pentru clienţi importanţi, care au comenzi mari.
Situaţia opusă este aceea în care furnizori mari vând unui număr mare de clienţi mai mici. În acest caz, forţa economică este de partea furnizorului, deci fiecare cumpărător trebuie să înveţe procedurile de catalogare, comandă şi taxare, unice pentru fiecare furnizor în parte. Clientul poartă povara respectării cerinţelor furnizorului şi nu are prea multă putere de a schimba ceva. Furnizorul nu va avea nici cea mai mică intenţie să îşi alinieze procedurile cu ceilalţi furnizori, deoarece în acest mod clientul poate să treacă mai uşor de la un furnizor la altul. Deci, o librărie care primeşte cărţi de la un furnizor important de cărţi, vrând-nevrând va utiliza sistemul de comenzi al furnizorului. Iar dacă veţi comanda o pereche de pantaloni la un mare magazin, nu veţi avea nici un succes dacă nu respectaţi sistemul acestuia.
Pe de altă parte, acolo unde există grupuri în domeniu, mai multe companii mici pot să se coalizeze într-un astfel de grup şi să creeze o schemă care să uşureze comerţul între membrii grupului şi furnizori. Dacă puterea de cumpărare este destul de mare pentru a putea să influenţeze şi furnizorii mai mari, poate să rezulte o relaţie de cooperare bazată pe standarde comune.

În multe domenii, informaţiile comerciale sunt o necesitate a activităţii în sine. Sănătatea, asigurările, băncile, agenţiile de credit, depanatorii auto, pieţele financiare şi chiar IRS-ul au interese proprii clare pentru încurajarea comunicării, deci nu este o surpriză că deja există standarde XML sau grupuri de studiu în toate aceste domenii. De fapt, principala barieră în calea noii tehnologii XML în unele domenii este că problemele de comunicare care au apărut au fost rezolvate o dată printr-o tehnologie mai veche care nu se bazează pe XML; toată lumea a adoptat soluţia respectivă, iar acum nimeni nu doreşte să ameninţe echilibrul prin trecerea la o alternativă nouă, mai bună. „Dacă nu este stricat, nu îl repara" este un principiu foarte important pentru mulţi responsabili în diverse instituţii şi corporaţii, iar dacă aceştia nu văd beneficii clare din trecerea la un nou format, nu vor fi de acord să se cheltuiască bani şi efort pentru schimbarea stării de fapt.
În schimb, la nivelul vânzării cu amănuntul, în care proprietarii de magazine se bat pentru banii fiecărui client, nu există astfel de interese. De fapt există chiar un dezinteres real. Chiar şi pe Web, unde preţurile trebuie să fie disponibile dacă o companie vrea să vândă ceva, afişarea preţurilor într-o manieră care să permită compararea efectuată de un motor de căutare automat şi nu de un client care a citit pagina cu reclame şi este gata să cumpere, l-ar pune pe gânduri şi pe cel mai destoinic vânzător. Deşi marile reduceri generale de preţuri pot fi folosite pentru a atrage clienţii la inaugurarea sitului, această abordare a dispărut încet din multe situri, accentul trecând de la preţul scăzut la comoditate şi scutirea de impozit pe profit.
Afişarea preţurilor pe Web într-un format uşor accesibil invită la cumpărarea prin comparaţii robotizate, iar majoritatea companiilor speră ca clienţii lor să nu facă niciodată aşa ceva. Singura companie care ar fi avantajată de comparaţiile de preţ este cea cu preţul cel mai mic. Toţi ceilalţi trebuie să ajungă la acel preţ, sau să îşi convingă clienţii că merită să cheltuiască mai mult. Aceste lucruri duc la o spirală în coborâre, pe care multe companii se străduiesc din greu să o combată prin reclame care accentuează valoarea unor lucruri intangibile – comoditate, fiabilitate – sau motive străine de obiect care să justifice cheltuirea a 50000$ pentru o maşină chiar dacă o bicicletă de 500$ ar fi de ajuns.

Notă Pe majoritatea siturilor de desfacere, singura modalitate de afişare a preţurilor este în reclame, deoarece nu sunt identificate de etichete XML. Cel puţin o companie importantă de vânzare on-line a început garantând că oferă cele mai mici preţuri, dar încet au retras afirmaţia pentru un mare număr de produse care nu se vindeau foarte bine, lăsând impresia clară dar falsă că pe situl lor se obţin cele mai bune preţuri. De fapt, destul de des aveau cel mai mare preţ pe Web, iar acum insistă asupra comodităţii şi a avantajului unui inventar masiv. Dacă aceste informaţii ar fi uşor de accesat de roboţi, aceste companii ar dispărea, deoarece oricine ar fi dotat cu un astfel de robot ar cumpăra de la cei care vând ieftin şi ar evita ofertele mai scumpe.

Deci, să nu vă aşteptaţi prea curând să vedeţi preţuri afişate de fiecare companie. Cei care le vor adopta primii vor fi firmele cu reduceri, care încearcă într-adevăr să ofere cel mai bun preţ şi să compenseze profitul redus prin volum mare de tranzacţii. Oricum, vor exista în continuare companii al căror renume cere să spună clientului, cel puţin figurativ, „Dacă doriţi să ştiţi cât costă, atunci nu vă puteţi permite."
Modelul cel mai obişnuit al siturilor Web este să prezinte mai întâi o pagină cu oferta, care vă prezintă motivele pentru care să cumpăraţi un produs, oricare ar fi acesta, iar apoi o legătură la o pagină de comandă, în care aflaţi în sfârşit şi preţul.

DTD-uri de succes

Secțiunea care urmează descrie unul dintre primele DTD-uri de succes, care a dat un impuls deosebit dezvoltării SGML și, în ultimă instanță, a pregătit scena pentru XML. Secțiunile următoare descriu câteva abordări ale aceleiași probleme, crearea și publicarea în cooperare a unor documente mari și complexe care seamănă destul de mult cu cărțile.

DTD-ul CALS

Unul dintre primele DTD-uri a fost o serie de standarde SGML promulgate de Departamentul Apărării SUA (Department of Defense – DOD) și denumite, cu obișnuita eficiență militară, CALS, Continuous Acquisition and Life-cycle Support system, Military Standard (MIL-STD) 196 et al (sistemul de suport pentru achiziții continue și ciclu de viață). CALS este proiectat pentru interoperabilitate cu alte standarde DOD, printre care FLIS, Federal Logistics Information System (sistemul federal pentru informații logistice), care conține date de descriere a tipurilor pentru mai mult de un milion de articole de dotare militară și piese de schimb, atât pentru forțele armate SUA cât și pentru mulți membri NATO, și National Stock Numbers (NSN) (numere de stoc naționale), care este un fel de Uniform Product Code pentru componente militare și rezerve. Există multe alte standarde militare SGML, pe care Pentagonul poate să le impună furnizorilor săi, și care guvernează procurarea tuturor materialelor, de la supă în conserve la cuirasate.

Notă Ca exemplu al complexității acestor sisteme, FLIS conține National Stock Number (NSN), numele standard al articolului, fabricanții autorizați, numerele de componente ale fabricantului, articolele care pot fi înlocuite, datele de încărcare și expediere, indicatori pentru materiale periculoase, caracteristici ecologice și de mediu, eficiența energetică, coduri pentru caracteristicile materialului, date de gestiune și caracteristici fizice și de performanță.

CALS a fost inițiat în 1984 și a fost creat pentru a ajuta armata să se descurce cu maldărele enorme de hârtii cerute de sistemele critice pentru misiuni și supraviețuire, care în armată tind să se suprapună. De atunci s-a dezvoltat și include multe standarde înrudite, printre care Joint Computer-aided Acquisition and Logistics Support system (JCALS) (sistemul reunit asistat de calculator pentru suportul achizițiilor și logisticii) ca infrastructură informațională, manager al fluxului operațional și ca manager global de date. JCALS este integrat cu Joint Engineering Data Management Information and Control System (JEDMICS) (sistemul reunit pentru ingineria datelor, gestionarea informației și control) pentru a stoca datele digitale și câteva aplicații în proprietatea guvernului, cum ar fi Configuration Management Information System (CMIS) (sistemul de configurare și management al informației) și Multi-user Engineering Change Proposal Automated Review System (MEARS) (sistemul multi-utilizator pentru inginerie, propuneri de modificare și revizuire automată). Liantul între toate aceste procese este un set de DTD-uri SGML care permit transferul continuu al datelor între sisteme.

Deoarece armata are cerințe de achiziții și suport care se întind pe decenii, și nu ani, un element cheie al succesului CALS este setul robust de instrumente asigurat de SGML și de DTD-urile CALS. Deoarece DTD-ul trebuie să satisfacă toate cerințele de achiziții

și aprovizionare ale întregului DOD, DTD-urile legate de CALS sunt foarte mari și numeroase. Acestea ajută acum la întreținerea sistemelor de aprovizionare ale armatei și vor continua să o facă mulți ani de acum înainte. Unele dintre DTD-urile CALS sunt traduse în XML, pentru a permite armatei să profite de noile standarde și instrumente care sunt disponibile, scăzând astfel costul procurărilor. Un sit al marinei care cuprinde multe informații, inclusiv DTD-urile Navy CALS, se află la adresa:

`http://navycals.dt.navy.mil/cals.html`.

Figura 26.1 vă prezintă un DTD al Navy CALS care este folosit pentru a edita un manual de operațiuni pentru Submarinul galben al celor de la Beatles în XMetaL 1.0, desigur o utilizare extrem de potrivită a acestui DTD. Textul este bineînțeles top secret și este hașurat.

Veți observa o asemănare foarte mare între acest DTD, care este creat pentru manuale navale, ISO-12083, Doc-Book și alte DTD-uri pentru documente asemănătoare cărților. Este destul de logic, deoarece comunitatea SGML a avut o influență foarte mare asupra XML; probleme similare au adesea ca rezultat scheme similare.

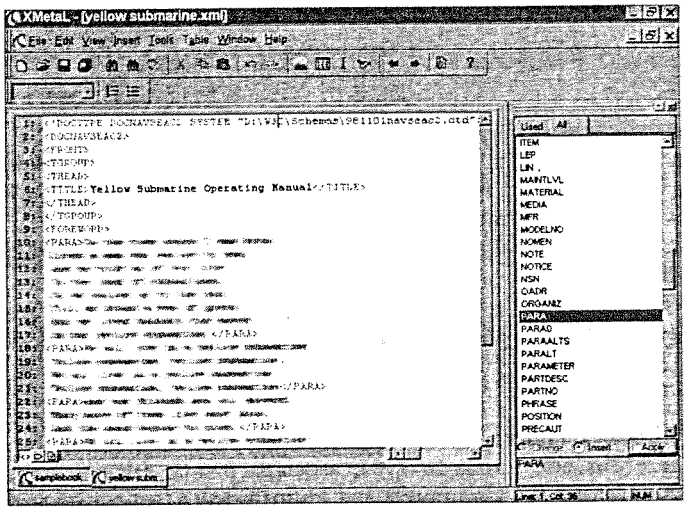


FIGURA 26.1
XMetaL 1.0 editează un manual de operațiuni pentru Submarinul galben.

ISO-12083 – Un DTD privat și public

Acest standard ilustrează una dintre problemele care apar la dezvoltarea unor DTD-uri private sau chiar semi-publice. Deși este suportat pe larg în industria editorială, acest DTD care descrie cărți și periodice nu este disponibil decât dacă plătiți la ISO privilegiul de a-l citi. Astfel se limitează dezbaterea liberă iar evaluările devin mai dificile, deoarece singurii care pot să vorbească în cunoștință de cauză sunt cei care au acces la standard. Dacă nu lucrați pentru o companie în care există un interes pentru ISO-12083, poate să fie destul de dificil să îi convingeți pe cei care trebuie să plătească, să scoată banii numai ca să îl studiați.

Unele versiuni schițe ale standardelor ISO-12083, refăcute în XML, sunt disponibile public la `http://www.xmlxperts.com/12083xml.htm`, dar specificația SGML completă este

disponibilă numai de la International Standards Organization (ISO), la http://www.iso.ch/.

Figura 26.2 vă prezintă o mică parte din încărcătura DTD-ului pentru cărţi ISO-12083 tradusă în XML ca schiţă de lucru. Deoarece sunt necesare aceleaşi elemente de bază ca în cazul DTD-ului pentru manuale al marinei, se aplică aceleaşi structuri generale de date: autor, titlu, capitole, paragrafe şi aşa mai departe. Puteţi vedea schiţele XML ale celor trei standarde ISO-12083 la situl XMLXperts, la http://www.xmlxperts.com/12083xml.htm.

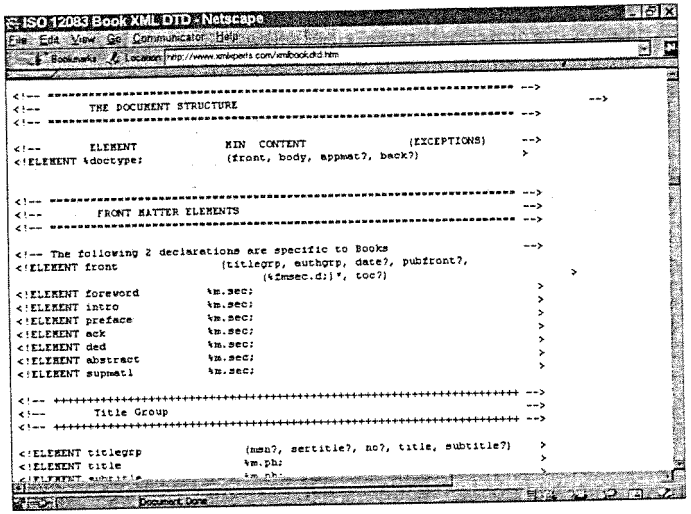


FIGURA 26.2
DTD-ul pentru cărţi ISO-12083 este prezentat aici parţial.

TEI – Text Encoding Initiative (iniţiativa pentru codificarea textului) pentru oamenii de ştiinţă

O altă abordare a fost a celor care au lucrat la Text Encoding Initiative. TEI s-a dezvoltat din dorinţa de a avea texte de bază disponibile într-o formă utilă pentru cercetarea ştiinţifică, fără a fi nevoie ca de câte ori apare un domeniu nou de cercetare să se inventeze din nou roata. În 1987, asociaţia pentru calculatoare şi ştiinţe umaniste (The Association for Computers and Humanities) a fost convocată la conferinţa pentru planificare de la Poughkeepsie pentru a studia posibilitatea creării unei scheme universale de codificare pentru marcarea textelor electronice, care să asigure o comunicare liberă şi să fie folosită în mediile educaţionale şi de cercetare. TEI este acum sub controlul unui consorţiu de organizaţii ştiinţifice şi se află la Universitatea Oxford.

Pagina de bază Text Encoding Initiative este la http://www.hcu.ox.ac.uk/TEI/ şi nu vă veţi plictisi aici cu detalii. Dacă doriţi să vedeţi DTD-urile, ar trebui să studiaţi sursa care oferă un ghid atât în formatul SGML cât şi în cel XML, şi vă arată cum să alcătuiţi o „pizza" din porţiuni modulare DTD pentru a construi un DTD particularizat.

DocBook – documentaţie portabilă

DocBook este o altă variaţiune pe tema DTD-ului pentru cărţi şi este păstrată de OASIS la http://www.oasis-open.org/docbook/, dar a fost iniţiată în 1991 de HaL Computer Systems şi O'Reilly & Associates. DocBook are un suport destul de larg în rândul autorilor şi editorilor şi este suportat direct de unele medii de creare SGML. Desigur, acest lucru se datorează şi faptului că este disponibil, deoarece multe DTD-uri sunt blocate în spatele drepturilor de autor şi a organismelor de standardizare care îşi finanţează activităţile din vânzarea rezultatelor muncii lor. Deoarece, la rândul meu, îmi câştig pâinea din scris, le împărtăşesc aceste dorinţe. Însă în cazul organismelor de standardizare, acest lucru presupune că uneori oamenii vor fi tentaţi să redescopere roata, doar ca să nu depindă de acestea. Actualizarea unui set de manuale poate fi o cheltuială importantă pentru o companie mică şi chiar în cazul unei companii mai mari, cu utilizatori numeroşi – fiecare dintre aceştia dorind să aibă un manual propriu – investiţia poate fi prea mare.

Notă OASIS, the Organization for the Advancement of Structured Information Standards (organizaţia pentru dezvoltarea standardelor de informaţie structurată) a apărut ca un depozit important de standarde XML şi promovează XML şi alte standarde în domeniu prin programe de educaţie şi cooperare. OASIS găzduieşte de asemenea şi XML.org.

MARC 21 – Machine Readable Cataloging (catalogare care poate fi citită de calculator)

The Library of Congress (Biblioteca Congresului) furnizează conţinut de catalogare şi denumire miilor de biblioteci locale, astfel încât are un standard de transfer al datelor, MARC 21, pe care îl sponsorizează împreună cu British Library (Biblioteca britanică), National Library of Canada (Biblioteca naţionala canadiană), The International Federation of Library Associations (IFLA – federaţia internaţională a asociaţiilor bibliotecilor) şi alţii. Deşi MARC nu este nici XML, nici SGML, mecanismul său de etichetare este uşor de tradus atât în SGML cât şi în XML, deci trebuie să fie menţionat aici, chiar şi numai ca un format EDI important, legat de publicaţii şi cărţi. Există deja o variantă XML simplă, MARCXML, de la Logos Research Systems, la http://www.logos.com/marc/.

Cu aceasta am încheiat discuţia detaliată cu privire la standardele de publicare legate de XML. În secţiunea următoare veţi vedea pe scurt alte consorţii din diferite domenii, cu explicaţii scurte şi generale despre tipul de comunicaţii facilitate de standardele promovate de fiecare organizaţie.

Alte grupuri legate de XML din diverse domenii

Secţiunile de mai sus nu epuizează subiectul domeniilor care cooperează în mod similar pentru îmbunătăţirea comunicării între diverse medii şi sisteme de operare. Lista următoare descrie pe scurt multe alte grupări:

- **Acord la http://www.acord.org/** – O grupare din domeniul asigurărilor care promovează formate XML comune pentru transferul de date între societăţile de asigurări. Acestea trebuie să comunice destul de mult între ele deoarece în acest fel pot să controleze fraudele. Altfel, dacă v-aţi asigura maşina la o duzină de societăţi aţi putea lua banii de la toate când este în mod misterios turtită de un cilindru compresor.

- Astronomical Markup Language (AML) la `http://pioneer.gsfc.nasa.gov/public/xml/` și la `http://monet.astro.uiuc.edu/~dguillau/these/` – Un limbaj de marcare care încurajează comunicarea datelor astronomice între observatoarele din întreaga lume. Deoarece orice observator poate să vadă doar o parte a cerului, observatoarele și astronomii trebuie să coopereze pentru o imagine completă a Universului.
- Astronomical Instrument Markup Language (AIML) la `http://pioneer.gsfc.nasa.gov/public/aiml/` și la `http://monet.astro.uiuc.edu/~dguillau/these/` – Un limbaj de marcare pentru instrumente astronomice.
- The Australia New Zealand Land Information Council (ANZLIC) Metadata la `http://www.erin.gov.au/database/metadata/anzmeta/` – Un limbaj de metadate cartografice (GIS – Geographic Information System) de la antipozi.
- Bank Internet Payment System (BIPS) la `http://www.fstc.org/projects/bips/` – Încă un sistem de plăți electronice și transfer de fonduri care se adresează pieței bancare.
- Bioinformatic Sequence Markup Language (BSML) la `http://www.visualgenomics.com/` – Un limbaj „bioinformatic" de marcare patentat. Bioinformatica se referă la datele strânse sau distribuite în scopul cercetării și informării despre genom. Acesta permite, printre altele, marcarea ușoară a informațiilor din secvențele ADN.
- Bio.Perl la `http://bio.perl.org/Projects/XML` – Un sit care dezvoltă instrumente de conversie dintr-un limbaj de marcare pentru biologie în altul, cum ar fi din BSML în BIOML sau invers.
- The BIOpolymer Markup Language (BIOML) la `http://www.proteometrics.com/BIOML/` – Încă un limbaj de marcare pentru biologie.
- Centre de recherche en droit public (CRDP) la `http://www.droit.umontreal.ca/crdp/en/equipes/technologie/index.html` – O bază de date legislativă canadiană în SGML.
- Chemical Markup Language (CML) la `http://www.xml-cml.org/` – Pagina de bază a CML.
- Coins: Tightly Coupled JavaBeans and XML Elements la `http://www.jxml.com/coins/index.html` – Situl unui furnizor de JavaXML.
- Cold Fusion Modeling Language (CFML) la `http://www.allaire.com/developer/index.cfm` – Limbajul de marcare Cold Fusion. Vizitați acest sit.
- CommerceNet Home Page la `http://www.commerce.net/` – Consorțiul de standarde pentru comerț electronic.
- Component Information Dictionary Specification (CIDS) la `http://www.si2.org/ecix/cids/index.html` – Un standard pentru componente electronice, dezvoltat de Electronic Component Information Exchange (ECIX).
- The Corpus Legis Project la `http://www.juridicum.su.se/iri/corpus/` – O bază de date legislativă suedeză.
- Customer Support Consortium la `http://www.customersupport.org/` – Un organism pentru standarde XML adresate organizațiilor pentru protecția consumatorilor.
- cXML, Commerce XML la `http://www.cxml.org/home/` – Un DTD pentru comerț electronic.
- Data Documentation Initiative la `http://www.icpsr.umich.edu/DDI/` – Un DTD XML pentru documentarea datelor.
- DDML/Xschema la `http://simonstl.com/xschema/` – O schemă XML alternativă sau o evoluție care pretinde că este mult mai bună decât XMLSchema.

- Digital Receipt Infrastructure Initiative la `http://www.receipt.org/` – Pagina de bază a Digital Receipt Consortium, o organizație pentru standarde de comerț electronic alcătuită din furnizori și instituții financiare.
- DISA, Data Interchange Standards Association la `http://www.disa.org/` – Un sit pentru standarde de comerț electronic.
- DMTF Common Information Model (CIM) la `http://www.dmtf.org/spec/index.html` – Pagina de standarde a Distributed Management Task Force, care descrie CIM, WBEM, DMI și standarde XML pentru consumatori.
- DMTF Desktop Management Interface (DMI) la `http://www.dmtf.org/spec/index.html` – Pagina de standarde a Distributed Management Task Force, care descrie CIM, WBEM, DMI și standarde XML pentru consumatori.
- DMTF Web-based Enterprise Management (WBEM) la `http://www.dmtf.org/spec/index.html` – Pagina de standarde a Distributed Management Task Force, care descrie CIM, WBEM, DMI și standarde XML pentru consumatori.
- ECIX – Electronic Component Information Exchange la `http://www.si2.org/ecix/` – Pagina de bază a QuickData (QuickData Protocol și Quick Evaluation Data Specifications), specificațiile PCIS, CIDS și TDML.
- Educom Instructional Management Systems Project (IMS) Metadata Specification la `http://www.imsproject.org/metadata/index.html` – Limbaj educațional de metadate bazat pe XML.
- EEMA EDI/EC Work Group: UN-XML la `http://www.edi-tie.nl/edifact/xml-edi.htm` – O propunere de depozit XML european, care va fi organizată sub auspiciile Națiunilor Unite.
- Encoded Archival Description (EAD) Home Page la `http://www.loc.gov/ead/ead.html` – Pagina de bază a unui standard SGML legat de MARC al Bibliotecii Congresului.
- EUROMATH Project la `http://stork.ukc.ac.uk/computer_science/EMT/` – Un editor de matematică european.
- European XML/EDI Pilot Project la `http://www.cenorm.be/isss/workshop/ec/xmledi/isss-xml.html` – Electronic Data Interchange dintr-o perspectivă europeană.
- Exchange Table Model la `http://www.oasis-open.org/html/publtext.htm` – O submulțime a modelului de tabele CALS pentru comunicațiile între furnizori.
- Frontier 5 and XML la `http://frontier.userland.com/stories/storyReader$1124` – Descrie instrumentele XML pe care le folosește un editor pentru a gestiona crearea și stocarea de conținut.
- GCA, Graphics Communications Association la `http://www.gca.org/` – Asociație de editori cu un interes puternic în standardele legate de poligrafie. Este sursa mai multor standarde, de interes minor pentru cei care nu sunt tipografi sau fabricanți de hârtie.
- GCA, Graphics Communications Association Paper DTD la `http://www.mulberrytech.com/MT99/mtxmldtd.html` – Un DTD al GCA pentru lucrările trimise spre publicare.
- GedML: [GEDCOM] Genealogical Data in XML la `http://home.iclweb.com/icl2/mhkay/gedml.html` – Un DTD XML genealogic.
- Georgia State University Electronic Court Filling Project la `http://gsulaw.gsu.edu/gsuecp/` – Un sistem din Georgia pentru documentele dintr-un tribunal.
- Health Level Seven la `http://www.hl7.org/` – Un grup din domeniul serviciilor de

sănătate care standardizează transferul fișelor medicale între furnizorii de servicii de sănătate.

- HR-XML Consortium la `http://www.hr-xml.org` – Standarde XML pentru departamentele de resurse umane.
- HTTP Distribution and Replication Protocol (DRP) la `http://www.w3.org/TR/NOTE-drp` – Un standard HTTP încă în dezvoltare.
- IEEESTD V3.0 DTD la `http://standards.ieee.org/resources/spasystem/dtd/index.html` – Standardul IEEE pentru standarde.
- Information & Content Exchange (ICE) la `http://www.w3.org/TR/NOTE-ice` – O propunere de protocol trimisă la W3C pentru publicarea simultană a conținutului. O parte a propunerii este DTD-ul complet ICE.
- Interactive Financial Exchange (IFX) la `http://www.IFXForum.org/` – Un standard XML pentru date financiare și procesarea acestora. Include standardul mai vechi Open Financial Exchange (OFX/OFE).
- JSML, Java Speech Markup Language la `http://java.sun.com/products/java-media/speech/` – Deși Sun mai promovează încă JSML, se presupune că acesta va fi înlocuit de limbajul Sable.
- Meta Content Framework (MCF) la `http://www.textuality.com/mcf/NOTE-MCF-XML.html` – Versiunea Netscape pentru RDF.
- Newspaper Association of America (NAA) – Standard for Classified Advertising Data la `http://www.naa.org/` – Pagina de bază a NAA. Vezi și `http://www.naa.org/technology/index.html`.
- Ontology Markup Language la `http://wave.eecs.wsu.edu/CKRMI/OML.html` – Sit ontologic care conține DTD-uri.
- Open Content Syndication (OCS) la `http://alchemy.openjava.org/ocs/` – Un format Netcenter Channel.
- OpenMLS, Real Estate DTD Design la `http://www.openmls.com/` – Limbaj de marcare pentru listări multiple din domeniul imobiliar.
- Open Software Description Format (OSD) la `http://www.marimba.com/products/whitepapers/osd-wp.html` – Propunerea Marimba/Microsoft către W3C pentru descrierea de software.
- Open Trading Protocol (OTP) la `http://www.otp.org/` – Pagina de bază a Consorțiului OTP, un grup de furnizori și instituții financiare care încearcă să standardizeze metode de comerț și plată electronică.
- OpenTag la `http://www.opentag.org/` – Un format comun de marcare pentru codificarea textului extras din documente de diferite formate.
- Platform for Internet Content Selection (PICS) la `http://www.w3.org/PICS/` – Un standard de metadate creat pentru a permite părinților să controleze siturile la care au acces copiii lor prin Internet.
- Process Interchange Format XML (PIF-XML) la `http://www.xmlns.com/dtd/pif.html` – O traducere XML a unui standard bazat pe Lisp.
- Rich Site Summary (RSS) la `http://my.netscape.com/publish/help/quickstart.html` – Pachetul de publicare simultană a conținutului, de la Netscape.
- RosettaNet la `http://www.rosettanet.org/` – Consorțiu de standarde pentru comerț electronic.

- Sable, Speech Synthesis Markup Language la `http://www.cstr.ed.ac.uk/projects/ssml.html` – Sable este un limbaj de marcare pentru sinteză vocală și succesorul lui SSML, JSML și STML.
- SAE J2008, Automotive Industry Standard la `http://www.xmlxperts.com/sae.htm` – Un standard XML pentru industria automobilistică. California cere introducerea înregistrărilor de întreținere SAE J2008 pentru toate mașinile noi vândute în acest stat începând cu 2002.
- Schema.net, The XML Schema Site la `http://www.schema.net/` – Un depozit de scheme XML.
- SIM SGML Database Technology, Royal Melbourne Institute of Technology (RMIT) la `http://www.mds.rmit.edu.au/sim_2.1/welcome.html` – Un gestionar de informații structurate din Australia.
- SmartX la `http://www.smartxml.com/` – Limbaj de marcare pentru cartele inteligente.
- SSML, Speech Synthesis Markup Language: Sable la `http://www.cstr.ed.ac.uk/projects/ssml.html` – Sable este un limbaj de marcare pentru sinteză vocală și succesorul lui SSML, JSML și STML.
- TalkML la `http://www.w3.org/Voice/TalkML` – Un limbaj XML experimental pentru browsere vocale, care folosește recunoașterea vocii.
- Telecommunications Interchange Markup (TCIF/IPI) la `http://www.atis.org/atis/tcif/index.htm` – Un consorțiu pentru standarde de telecomunicații. Căutați în comitete pentru activitățile TCIF legate de comerțul electronic.
- Theological Markup Language (ThML) la `http://www.ccel.org/` – Un limbaj de marcare teologic cu etichete speciale pentru referințe la scripturi, numere Strong's Exhaustive Concordance și altele.
- Translation Memory eXchange la `http://www.lisa.org/tmx/index.html` – Un DTD pentru schimbul de date de translatare a memoriei între instrumente de translatare sau furnizori, cu pierderi de date minime sau zero.
- UML eXchange Format (UXF) la `http://www.yy.cs.keio.ac.jp/~suzuki/project/uxf` – Format XML pentru schimb de date pentru modele în Unified Modeling Language.
- UIML, User Interface Markup Language la `http://uiml.org/` – Un limbaj de marcare pentru interfețe cu utilizatorul.
- Virtual Hyperglossary (VHG) la `http://www.vhg.org.uk/` – Un glosar de termeni din Regatul Unit folosiți pe Web.
- VoiceXML la `http://www.alphaworks.ibm.com/tech/voicexml` – Limbaj de recunoaștere a vocii al IBM.
- Weather Observation Markup Format (OMF) la `http://zowie.metnet.navy.mil/~spawar/JMV-TNG/` – Pagina de bază a limbajului de marcare meteorologic.
- Web Standards Project (WSP) la `http://www.webstandards.org/` – Un grup de utilizatori care încearcă să standardizeze browserele. Acum pun accentul pe CSS, o bătălie curajoasă, chiar dacă este inutilă, în opinia mea.
- Web-based Distributed Authoring and Versioning (WEBDAV) la `http://www.ics.uci.edu/pub/ietf/webdav/` - Un DTD proiectat pentru documente de colaborare, care permite autorilor să urmărească activitatea celorlalți.
- Wireless Markup Language (WML) la `http://www.wapforum.org/` – Limbaj de marcare pentru telefoane mobile, asistenți digitali și pagere.

- **XFDL, Extensible Forms Description Language** la `http://www.uwi.com/xfdl/` – Un limbaj de descriere pentru formulare, care permite, printre altele, şi semnarea formularului.
- **XLF (Extensible Log Format) Initiative** la `http://www.docuverse.com/xlf` – Format de înregistrare pentru servere bazate pe XML.
- **XML Court Interface (XCI)** la `http://www.nmcourt.fed.us/xci/xcihome.htm` – Un sistem de documente XML pentru tribunale, din New Mexico.
- **XML/EDI Group** la `http://www.geocities.com/WallStreet/Floor/5815/` – O pagină Geocities (mai trebuie spus ceva?) care descrie un standard pentru schimbul de date electronice. Lista susţinătorilor este surprinzător de lungă.
- **XML/EDI Repository** la `http://www.xmledi.com/repository/` – Un sit mai credibil al grupului Geocities XML/EDI menţionat mai sus. Intenţia este să încurajeze dezvoltarea de DTD-uri în domeniul public şi să le facă disponibile gratuit pe Web.
- **XMLNews** la `http://www.xmlnews.org/` – DTD-uri XML pentru ştiri.
- **XML NITF** la `http://www.iptc.org/iptc/` şi `http://www.nitf.org/` – DTD-ul News Industry Text Format.
- **XML-RPC Home Page** la `http://www.xmlrpc.com` – Apelări la distanţă ale procedurilor mediate prin XML!
- **Simple Object Access Protocol (SOAP)** la `http://www.ietf.org/internet-drafts/draft-box-http-soap-01.txt` – Un alt limbaj pentru apelul la distanţă al procedurilor, creat de Dave Winer.

Arhive DTD şi organisme de standardizare

Următoarele referinţe reprezintă standarde sau propuneri de standarde de la W3C şi de la alte câteva organisme de standardizare. De departe, cele mai multe provin de la W3C, deoarece acest organism este răspunzător pentru o mare parte din realizările XML, dar sunt reprezentate şi International Standards Organization (ISO) şi alte organizaţii.

Notă Deşi descrierile listate aici sunt exacte pentru momentul scrierii acestei cărţi, propunerile de recomandări şi, de asemenea, unele dintre schiţele de lucru din listă pot fi Recomandări în momentul în care citiţi aceste rânduri. În acest caz, locaţiile lor pe Web s-au schimbat. Vă rog să consultaţi situl de bază al W3C, la `http://www.w3.org/` pentru informaţii actualizate.

- **Annotated XML Specification** la `http://www.xml.com/axml/axml.html` – Recomandarea foarte bine scrisă a XML 1.0, adnotată de Tim Bray.
- **APPEL** la `http://www.w3.org/TR/WD-P3P-preferences` – O schiţă de lucru W3C care descrie o abordare a P3P.
- **Associating Style Sheets with XML documents Version 1.0** la `http://www.w3.org/TR/xml-stylesheet` – O recomandare W3C care descrie un mod standard de a asocia o foaie de stil cu un document.
- **CSS1, Cascading Style Sheets level 1** la `http://www.w3.org/TR/REC-CSS1` – Recomandarea revizuită CSS1 de la W3C.
- **CSS2, Cascading Style Sheets level 2** la `http://www.w3.org/TR/REC-CSS2` – Recomandarea CSS2 de la W3C.
- **Canonical XML** la `http://www.w3.org/TR/xml-c14n` – O schiţă de lucru W3C care descrie un format canonic XML, folosit pentru a compara documentele XML pentru diferenţe sau echivalenţe.
- **Common Markup for micropayment per-fee-links** la `http://www.w3.org/TR/WD-Micropayment-Markup/` – O schiţă de lucru W3C.
- **DOM Level 1** la `http://www.w3.org/TR/REC-DOM-Level-1/` – Recomandarea DOM Level 1 de la W3C.
- **DOM Level 2** la `http://www.w3.org/TR/WD-DOM-Level-2/` – Schiţa de lucru DOM Level 2 de la W3C.
- **Extensible Stylesheet Language (XSL) Specification** la `http://www.w3.org/TR/WD-xsl` – Schiţa de lucru XSL de la W3C.
- **HTML 4.0** la `http://www.w3.org/TR/REC-html40/` – Recomandarea HTML 4.0 de la W3C.
- **HTML 4.01** la `http://www.w3.org/TR/html40/` – Propunerea de recomandare HTML 4.01, care adaugă mai multe opţiuni multimedia, limbaje de script, foi de stil, facilităţi de tipărire mai bune, accesibilitate mai bună pentru persoanele cu handicap şi îmbunătăţiri de internaţionalizare.
- **ISO/IEC FCD 13250:1999 – Topic Maps** la `http://www.ornl.gov/sgml/sc34/document/0058.htm` – Un standard propus de ISO, care descrie o metodă de a pune în corespondenţă resursele folosite pentru a defini un subiect.
- **International Layout in CSS** la `http://www.w3.org/TR/WD-i18n-format/` – O schiţă de lucru W3C care descrie cerinţe de internaţionalizare suplimentare (i18n) pentru tipărirea unor limbi asiatice (coreeană, japoneză, chineză) şi bidirecţionale.
- **MathML (Mathematical Markup Language)** la `http://www.w3.org/TR/REC-MathML/` – Recomandarea W3C care descrie MathML.
- **Namespaces in XML** la `http://www.w3.org/TR/REC-xml-names/` – Recomandarea pentru zone de nume de la W3C.
- **P3P, Platform for Privacy Preferences** la `http://www.w3.org/TR/wd-p3p` – Schiţa de lucru W3C pentru P3P.
- **PICS Signed Labels** la `http://www.w3.org/TR/REC-DSig-label/` – Recomandarea Platform for Internet Content Selection Signed Labels.
- **Paged Media Properties for CSS3** la `http://www.w3.org/TR/css3-page` – Schiţa de lucru W3C care descrie extinderile propuse la standardele CSS.
- **RDF Model and Syntax** la `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/` – Recomandarea W3C pentru RDF.
- **RDF Schema** la `http://www.w3.org/TR/PR-rdf-schema/` – Propunerea de recomandare a W3C pentru RDF Schema.
- **RFC-2112, MIME Multipart/Related** la `http://www.faqs.org/rfcs/rfc2112.html` – RFC Internet care înlocuieşte RFC-1872 mai vechi, în legătură cu documentele Multipart MIME.
- **Ruby** la `http://www.w3.org/TR/WD-ruby` – Schiţa de lucru W3C care descrie suportul pentru text „ruby", necesar în chineză şi în alte limbi asiatice. Termenul „ruby" se referă de fapt la mărimea de caractere de 5.5 puncte şi, prin extensie, la orice dimensiune mică de caractere. Textul ruby este folosit pentru a indica pronunţia sau alte adnotări pe lângă un cuvânt de bază sau o frază, utilizarea fiind dedusă din context.

- **SMIL, Synchronized Multimedia Integration Language** la `http://www.w3.org/TR/REC-smil/` – Recomandarea W3C pentru SMIL.
- **SMIL Boston** la `http://www.w3.org/TR/SMIL-Boston` – Schița de lucru W3C care descrie următorul nivel SMIL.
- **SVG, Scalable Vector Graphics** la `http://www.w3.org/TR/SVG/` – Schița de lucru W3C pentru SVG.
- **SVG Requirements** la `http://www.w3.org/TR/WD-SVGReq` – Schița de lucru W3C pentru SVG Requirements.
- **String Identity/Matching** la `http://www.w3.org/TR/WD-charreq` – Schița de lucru W3C care descrie cerințele pentru indexarea și compararea șirurilor.
- **Unicode in XML and other Markup Languages** la `http://www.w3.org/TR/Unicode-xml` – Schița de lucru W3C pentru utilizarea Unicode în XML.
- **User Interface for CSS3** la `http://www.w3.org/TR/css3-userint` – Schița de lucru W3C care descrie interfața cu utilizatorul a CSS3.
- **VoiceXML** la `http://www.voicexml.org/` – Un alt limbaj de marcare pentru sinteză vocală.
- **WAI, Web Accessibility Initiative** la `http://www.w3.org/WAI/` – Pagina de bază a WAI de la W3C.
- **WAI Authoring Tools Guidelines** la `http://www.w3.org/TR/WAI-AUTOOLS/` – Propunerea de recomandare W3C care descrie cerințele pentru instrumentele de creație WAI.
- **Web Content Accessibility Guidelines 1.0** la `http://www.w3.org/TR/WAI-WEBCONTENT/` – Recomandarea W3C pentru WAI Content Guidelines.
- **User Agent Accessibility Guidelines 1.0** la `http://www.w3.org/TR/WAI-USERAGENT/` – Schița de lucru W3C pentru WAI User Agent Guidelines.
- **Web Characterization** la `http://www.w3.org/TR/NOTE-WCA/` – O activitate a W3C pentru colectarea de date statistice despre utilizarea Web în HTTP-NG.
- **WebCGM Profile** la `http://www.w3.org/TR/REC-WebCGM/` – Recomandarea W3C pentru WebCGM, Computer Graphics Metafile.
- **XHTML 1.0** la `http://www.w3.org/TR/xhtml1/` – Propunerea de recomandare XHTML 1.0 a W3C.
- **XML 1.0** la `http://www.w3.org/TR/REC-xml/` – Recomandarea XML 1.0 a W3C.
- **XML Canonicalization Requirements** la `http://www.w3.org/TR/NOTE-xml-canonical-req` – Nota W3C care descrie cerințele pentru forma canonică a XML.
- **XML Fragment Interchange** la `http://www.w3.org/TR/WD-xml-fragment` – Schița de lucru W3C pentru schimbul de fragmente XML.
- **XML Information Set** la `http://www.w3.org/TR/xml-infoset` – Schița de lucru W3C care descrie informația disponibilă într-un document XML.
- **XML Path Language (XPath) Version 1.0** la `http://www.w3.org/TR/xpath` – Propunerea de recomandare W3C pentru XPath.
- **XML Pointer Language (XPointer)** la `http://www.w3.org/TR/WD-xptr` – Schița de lucru W3C pentru XPointer.
- **XML Schema Part 1: Structures** la `http://www.w3.org/TR/xmlschema-1/` – Schița de lucru pentru structurile XML Schema.

- **XML Schema Part 2: Datatypes** la `http://www.w3.org/TR/xmlschema-2/` – Schița de lucru pentru tipurile de date XML Schema.
- **XML Signature** la `http://www.w3.org/Signature/` – Pagina de bază a XML Signature de la W3C.
- **XLink** la `http://www.w3.org/TR/WD-xlink` – Schița de lucru W3C pentru XLink.
- **XML.org** la `http://www.xml.org/` – Un portal în domeniu, care oferă în principal o listă a organismelor de standardizare care creează efectiv standarde XML.
- **XSL, Extensible Stylesheet Language Specification** la `http://www.w3.org/TR/WD-xsl/` – Schița de lucru W3C pentru XSL.
- **XSLT, XSL Transformations Version 1.0** la `http://www.w3.org/TR/xslt/` – Recomandarea XSLT a W3C.

Software legat de XML

Siturile următoare oferă resurse importante pentru creatorii XML și proiectanții Web. Totul, de la instrumente XML până la seturi de testare și îndrumare, apare și aproape toate siturile oferă bogății neașteptate exploratorului sârguincios.

- **Academia Sinica Computing Centre** la `http://www.ascc.net/` – Pagina de bază a Academia Sinica Computing Centre. Găzduiește Chinese XML Now!
- **Chinese XML Now! Site** la `http://www.ascc.net/xml/` – Resursa definitorie pentru programatorii care doresc să testeze sau să dezvolte aplicații în XML în chineză. Figura 26.3 vă prezintă unul dintre fișierele lor de test, codat în chineză în Microsoft Internet Explorer. Există aici un set de testare pas cu pas care va fi util oricărui dezvoltator de XML.

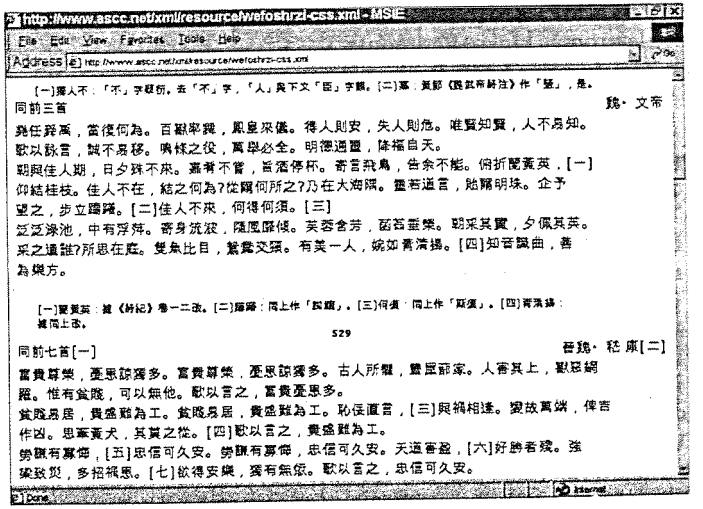


FIGURA 26.3
Acesta este unul dintre numeroasele fișiere excelente de testare de pe situl Web Chinese XML Now! găzduit de Academia Sinica Computing Centre.

- **developerWORKS, IBM** la `http://www.ibm.com/developer/xml/` – Situl cu resurse pentru programatorii XML al IBM.

- **GNOME, Gnu Network Object Model Environment** la `http://www.gnome.org/` – Sursa Gnu pentru software XML gratuit, printre care dia, un instrument de întocmit diagrame; go, un editor; gXaio, un organizator de album de imagini în XML; tim, un browser Web care suportă limbaje XML selectate și multe altele.
- **Free XML software** la `http://www.garshol.priv.no/download/xmltools/` – O colecție de software XML de Lars Marius Garshol din Norvegia.
- **Microsoft Internet Explorer** la `http://msdn.Microsoft.com/xml/` – Situl general XML al Microsoft.
- **Mozilla** la `http://www.mozilla.org/rdf/doc/` – Planurile legate de RDF și XML ale Mozilla.org.
- **Python** la `http://www.python.org/` – Analizoare și altele în Python.
- **XMLephant** la `http://xmlephant.com/` – Un sit de resurse pentru programatori. Se laudă că este „the BIG XML Resource" (cea mai mare resursă XML).
- **XMLSOFTWARE** la `http://xmlsoftware.com/` – Un sit de resurse pentru programatori.
- **XML Toolkit** la `http://csmctmto.interpoint.net/didx/xml.html` – Un instrument Python pentru dezvoltarea de cod XML.
- **Café Con Leche: XML News and Resources** la `http://metalab.unc.edu/xml/` – Un sit de știri și resurse legate de XML.
- **XML Hack: Developer news from the XML Community** la `http://www.xmlhack.com/` – Un sit de știri pentru comunitatea de programatori XML.
- **DTD.com: XML-based Language Standards Center** la `http://www.dtd.com/` – Un centru pentru standarde de limbaje bazate pe XML, cu o arhivă cuprinzătoare de DTD-uri XML.
- **Perl XML: XML news for Perl developers** la `http://www.perlxml.com/` – Situl revistei electronice lunare Perlmonth.
- **The Apache XML Project** la `http://xml.apache.org/` – Pagina de bază a Cocoon, un editor pentru Web bazat pe XML; Xalan, un procesor pentru foi de stil XSLT; Xerces, analizoare XML și FOP, un analizor pentru obiecte de formatare în XSL.

Studii de caz

Din bogăția de referințe și legături pe care le-ați putut vedea în acest capitol, vă dați seama cu ușurință că activitatea s-a întețit în ultimii ani, de la publicarea Recomandării W3C pentru XML 1.0. De fapt sunt atâtea, încât este greu să decideți ce este relevant și ce nu pentru un anumit subiect. Ca regulă generală, legăturile furnizate în pagina de bază a W3C legate de un anumit subiect XML sunt actuale și se lucrează activ la ele, dar sunt și alți autori și grupuri independente care dezvoltă vocabulare XML în numeroase domenii.

Va trebui să judecați cu discernământ atunci când vă decideți să urmați o cale sau un standard din domeniu, sau să dezvoltați propriul standard care să se potrivească perfect scopului dumneavoastră. În cazul din urmă, studiul unor lucrări pe același subiect vă poate stimula creativitatea sau vă ferește de greșelile și lacunele care sunt prezente în standardele sau instrumentele existente.

XML încurajează atât experimentul cât și evoluția, transparența și ușurința în utilizare a

vocabularelor XML, înlăturând numeroase bariere din calea dezvoltării, iar metodele simple prin care pot fi extinse sau modificate vocabularele existente permit folosirea rezultatelor anterioare ca bază pentru dezvoltările noi.

▶ Pentru o privire detaliată asupra extinderii unui DTD existent **vezi** „Extinderea unui DTD", la **pagina 93.**

Dar având în vedere dezvoltările care au loc în atâtea domenii, aproape sigur există deja rezultate disponibile în aproape orice domeniu. Printr-o căutare pe Web în domeniul care vă interesează, veți afla rezultate surprinzătoare. O căutare a combinației dintre XML și un cuvânt luat la întâmplare care vă vine în minte, de exemplu „etymology" (etimologie) a dat 72 de rezultate în motorul de căutare Altavista. Cel puțin o parte din acestea sugerează că există deja DTD-uri care conțin elemente etimologice pentru editorii de dicționare și enciclopedii, sau măcar sunt în curs de apariție. Ar fi o prostie să nu continuați căutarea mai departe, dacă proiectul dumneavoastră presupune informații etimologice, ceea ce bineînțeles este destul de improbabil. Dar o căutare similară, folosind cuvinte cheie importante în domeniul dumneavoastră de activitate, combinate cu cuvântul „XML" sau „SGML" poate să producă referințe valoroase.

Concluzii – viitorul în Web

CAPITOLUL 27

În acest capitol

- Coregrafia XML
- Puterea în mâinile utilizatorului
- Mai multe limbi
- Medii multiple
- Fluxuri de date multiple
- În serviciul mai multor tipuri de utilizatori
- Viitorul HTML
- Alternativele viitorului

Coregrafia XML

Într-o oarecare măsură, știți ce se *plănuiește* pentru XML, deoarece W3C a publicat luări de poziție care descriu direcția în care văd *ei* că se va îndrepta Web-ul în viitor. Această direcție, desigur, are XML în rolul principal. Deoarece XML este destinat să fie accesibil oricând, oriunde și pe orice dispozitiv capabil să ajungă pe Web, va trebui ca proiectanții Web să-și schimbe puțin modul de gândire înainte ca *dumneavoastră* să fiți la fel de pregătit pentru viitor ca și XML.

Printre talentele necesare este și abilitatea de a acționa ca un coregraf pentru informații, punând în scenă o prezentare completă prin care paginile noastre să poată fi adaptate interactiv la fluxul unui mediu Web în continuă schimbare. Dar XML este încă în evoluție, iar dacă luăm ca exemplu evoluția HTML, nimeni nu știe exact ce va urma.

Starea actuală a XML și a standardelor înrudite ne amintește de primele zile ale cinematografiei sonore. Mulți au presupus că noua tehnologie va schimba lucrurile, dar nimeni nu putea să prevadă ultima versiune a filmului *Godzilla*, plină de efecte speciale de vârf, după ce a văzut primele filme sonore. La fel, *este* clar că „insulele de date" XML – folosite de Microsoft pentru a îngloba fragmente XML în documente HTML – nu sunt o soluție pe termen lung. Orice soluție provizorie care presupune ascunderea documentului de bază XML sau fragmentarea sa în bucăți de dimensiunea unui octet este o sfidare din partea autorului său. Ascunderea documentului XML prin transformarea sa într-un fel de HTML presupune că autorul transformării știe deja ce vrea să facă fiecare utilizator cu informația conținută în documentul XML original și poate să ofere o soluție HTML care se potrivește pentru toate cerințele, caz în care XML este inutil din primul moment, dar noi știm deja mai bine care este situația.

Nici un individ sau grup nu poate să prevadă ce va dori altul să facă cu informațiile sau cum va utiliza acesta informațiile publicate pentru a face noi descoperiri sau invenții. Progresul științific izvorăște din circulația liberă a ideilor. Progresul pe Web va apărea în pofida filtrelor de date și transformărilor, și nu din cauza acestora.

Transformarea unui document de bază XML în HTML pentru a fi redat pe Web trebuie să ia în considerare cel mai mic numitor comun, browserul HTML moștenit. Din nefericire, se renunță în același timp și la multe din avantajele pe care le are XML. Deși asigurarea informației la cel mai mare număr de utilizatori poate să ceară echivalentul HTML al unui abecedar pentru copii, pentru aceia care nu sunt pregătiți pentru mai mult, nimeni nu va spune vreodată că un abecedar este mai bun decât realitatea. Deși o mare parte din discuțiile actuale despre XML se concentrează asupra plasării controlului în mâinile autorului, vine o vreme în care chiar și cel mai inteligent autor trebuie să cedeze controlul. Un autor furnizează sau chiar organizează informația, iar utilizatorii vor putea să facă ce vor dori cu ea, acesta este viitorul, nu scheme inalienabile care lasă pentru totdeauna în seama autorului decizia asupra modului în care se vor folosi informațiile.

Puterea în mâinile utilizatorului

Unul dintre avantajele reale ale XML este că pune puterea în mâinile utilizatorului, sau a unui agent utilizator inteligent, și eliberează utilizatorul care poate să definească tipul de publicație pe care îl vrea și nu caută prin rafturi niște informații preambalate care *s-ar potrivi* cu ceea ce dorea. Deci utilizatorii viitorului vor fi *constructori de informații* și *consumatori de informații*, iar dihotomia dintre aceștia va fi la fel de clară că și cea dintre cei care *joacă* fotbal și cei care *se uită* la fotbal. Un constructor de informații va proiecta și va coordona construcția documentului Web așa cum un arhitect proiectează și coordonează construcția unei clădiri, reunind materiale și tehnici disparate într-un tot coerent și elocvent.

Un consumator de informații poate fi oricine, de la un cercetător științific care efectuează o căutare literară exhaustivă la un adolescent care caută ultimele noutăți muzicale și video doar ca să se distreze.

Nu va dori toată lumea să învețe să genereze publicații XML interactive, și este clar că instrumentele actuale nici măcar nu permit așa ceva, majoritatea fiind înghețate în modelele rigide ale cerințelor lipsite de imaginație ale marilor corporații, care ignoră visele artiștilor și inventatorilor. Dar creativitatea va duce în cele din urmă la apariția unor versiuni reale ale dispozitivelor care acum sunt doar science-fiction.

Întrebări vocale adresate unui „computer" ca în *Star Trek* sunt doar începutul posibilităților. O instrucțiune aleatoare cum ar fi „Calculează relația statistică dintre grupa de sânge și mâna dominantă și trasează un grafic care arată variația acestei relații în funcție de distanță și de direcția cercului mare de la Denali" ar putea duce la o altă întrebare care să ducă la un medicament pentru vindecarea răcelii sau la un motor warp funcțional pentru navele spațiale. Cine știe?

Știm că informația din fișierele HTML este o masă omogenă aproape inutilă, iar informațiile din bazele de date patentate sunt aproape inaccesibile din exterior. Fără XML sau ceva atât de asemănător încât nu poate fi deosebit de acesta, nu puteți pune întrebările potențial cruciale fără o muncă prealabilă vastă și costisitoare de colectare și pregătire a

datelor. Dacă veţi putea merge vreodată la un calculator, să vorbiţi cu el şi să vă aşteptaţi să vă răspundă la întrebarea pusă într-o limbă naturală nu este foarte clar, dar utilizatorii încep să pună deja astfel de întrebări de la o tastatură.

Notă

Recunoaşterea vocii într-o limbă naturală aşteaptă în primul rând o îmbunătăţire cu câteva ordine de mărime a memoriilor şi a vitezei de procesare pentru a aproxima capacitatea creierului uman. Mulţi cred acum că viteze de procesare de 1000 MHz şi memorii de circa 1GB vor permite cel puţin un nivel primitiv de recunoaştere a vocii prin învăţare, dar recunoaşterea completă, independentă de vorbitor, este încă de neatins şi pentru următorii ani. Dar peste zece ani, poate că tastaturile vor fi la fel de demodate ca şi bicele căruţaşilor sau furcile de lemn. Poate le veţi vedea atârnate pe pereţi în restaurantele şic.

Ce este sigur este că nimeni nu ar fi găsit răspunsul la acea întrebare, sau la multe altele, de la început; răspunsul va fi găsit prin căutarea în datele brute, prin alipirea informaţiilor din mii de surse diferite, prin sintetizarea ordinii din haos şi extinderea cunoaşterii dintr-o ignoranţă relativă.

XML este liantul care uneşte toate aceste surse de date. Limbajele bazate pe XML descriu şi identifică datele, în timp ce foile de stil descriu opţiunile de redare disponibile pentru diferite dispozitive şi diferiţi utilizatori.

Mai multe limbi

Un lucru care poate se va schimba în viitorul apropiat este dependenţa de o limbă a sitului, o limbă a browserului şi o limbă a utilizatorului. Internaţionalizarea (I18N) este deja o parte componentă a filozofiei XML, cu îmbunătăţiri substanţiale faţă de schemele I18N mai vechi, care cereau un sistem de operare localizat pentru ca suportul pentru limbi străine să fie posibil. Mozilla – personificarea open-source a lui Netscape Navigator, finanţată în principal de Netscape, dar cu contribuţii semnificative şi din alte părţi ale comunităţii utilizatorilor – o ia foarte în serios, iar Microsoft Internet Explorer are deja un număr mare de module pentru suportul lingvistic. Dificultatea era până acum faptul că sistemele de operare mai vechi, cum sunt versiunile Windows anterioare lui Windows 2000 şi Windows NT, nu suportă nativ seturile de caractere ISO/Unicode. Windows 95 şi versiunile mai recente permit aplicaţiilor să asigure pe cont propriu suportul Unicode, dar limitările de memorie de pe majoritatea calculatoarelor mai vechi, cu Windows, fac ca această soluţie să fie destul de problematică. Deşi este destul de simplu să daţi posibilitatea unui utilizator să vadă un sit folosind un anumit set de caractere, posibilitatea interactivităţii (cum ar fi completarea unui formular) în limba respectivă este o cu totul altă problemă, în afara cazului în care aplicaţia a fost scrisă special pentru un suport Unicode.

Localizarea

Localizarea (L10N) este capacitatea de a modifica interfaţa cu utilizatorul pentru a se potrivi cu limba maternă (sau cea mai comodă) a unui utilizator. Microsoft are o acoperire excelentă a limbilor „mari", dar Mozilla este interesant prin acoperirea planificată a limbilor „mici". Este o măsură a devotamentului comunităţilor de utilizatori faptul că unele persoane s-au angajat să producă interfeţe utilizator pentru atât de multe

limbi vorbite de populaţii relativ mici. Browserul Opera al Opera Software din Norvegia acoperă bine limbile europene dar oferă un suport relativ restrâns pentru limbile din afara comunităţii europene. Dacă limba este un criteriu important pentru dumneavoastră, atunci ar trebui să aruncaţi o privire asupra suportului actual pentru limbi, înainte de a achiziţiona un produs.

Figura 27.1 vă prezintă ziarul egiptean *El Shaab* în arabă. Deoarece suportul browserelor pentru alfabetul arab este foarte slab, întregul sit este alcătuit din imagini grafice. Deci singura modalitate de a tăia şi lipi un citat este cu creionul şi hârtia!

FIGURA 27.1
Vorbitorii de arabă merită browsere care ştiu în mod nativ să redea araba şi baze de date care pot fi interogate în limba proprie.

Suportul lingvistic în trei browsere

Internaţionalizarea Web-ului va continua în mod sigur cu sau fără ajutorul W3C, dar iniţiativa lor I18N asigură energia din spatele multor eforturi, nu ultimul dintre acestea fiind suportul foarte încurajator pentru I18N al producătorilor de browsere. Înainte de efortul de standardizare al W3C, suportul lingvistic cerea un amestec încurcat de fonturi patentate, fiecare cu propriile standarde de codificare. Deşi au existat browsere speciale pentru multe limbi, cum ar fi browserul Sindbad de la Sakhr pentru arabă la `http://www.sakhr.com`, fiecare avea propriile limitări şi idiosincrazii. Nu exista o soluţie unică.

Printre browserele importante nici unul nu asigură suport pentru toate limbile. Sunt suportate doar limbile cu destui utilizatori potenţiali, pentru ca efortul să fie justificat din punctul de vedere al producătorilor.

Limbile Mozilla (Netscape)

Mozilla este în mod sigur în fruntea celor care asigură suport pentru limbi la care nimeni nu s-ar gândi să le suporte, deoarece pur şi simplu nu există baza economică pentru a justifica investiţia în dezvoltare. Dar una dintre cele mai puternice caracteristici ale filozofiei open-software este că aproape orice poate şi va fi suportat dacă cineva doreşte să o facă.

Dar cealaltă parte a problemei open-source este că o limbă nu va fi suportată dacă nu se oferă nimeni să o facă, deci suportul lingvistic inițial al Mozilla va fi destul de neregulat. Însă Mozilla diferă de majoritatea proiectelor open-source, deoarece are în spate ca finanțator o mare corporație, deci dacă suportul pentru o limbă se justifică din punct de vedere economic, atunci va fi asigurat în cele din urmă.

Netscape va asigura fără îndoială suportul pentru limbile din baza lor actuală, printre care sunt limbile cehă, chineză simplificată, chineză tradițională, coreeană, daneză, engleză (americană), engleză (britanică), finlandeză, franceză, franceză non-SMIME, germană, greacă, italiană, japoneză, maghiară, norvegiană, olandeză, poloneză, portugheza braziliană, rusă, slovenă, spaniolă, suedeză și turcă. Destul de ciudată este absența din browserul actual a arabei și ebraicii, care reprezintă concentrări majore de putere economică și/sau politică. De asemenea, lipsește acoperirea sub-continentului indian, o altă grupare lingvistică și culturală majoră care în mod sigur va fi extrem de importantă în viitor.

Mozilla este numele de proiect al generației următoare a browserului Netscape, cu un concept open-source și suportat cu ajutorul comunității utilizatorilor. Mozilla a fost un nume Netscape încă de la început și este folosit din nou aici din motive atât culturale, cât și istorice.

Printre limbile care vă puteți aștepta să fie suportate în Mozilla sunt:

- Bosniacă
- Bulgară
- Catalană
- Cehă
- Chineză simplificată
- Chineză tradițională
- Daneză
- Ebraică
- Engleză
- Franceză
- Galeză
- Germană
- Greacă
- Gruzină
- Havaiană
- Indoneziană
- Italiană
- Japoneză
- Maghiară
- Malaieză
- Norvegiană (Nynorsk și Bokmaș)
- Olandeză
- Poloneză
- Portugheză braziliană
- Română
- Rusă

- Suedeză
- Thailandeză
- Turcă
- Ucraineană

Limbile Microsoft Internet Explorer

Microsoft Internet Explorer acoperă din punct de vedere al populației și puterii economice cele mai multe din pietele importante ale lumii. Ca lărgime sunt în mod sigur premianți, deoarece fiecare grup „important" de limbi este bine reprezentat. În schimb, dacă limba dumneavoastră maternă nu face parte dintr-o categorie importantă, este improbabil să fie adăugată curând în listă. Limbile suportate sunt:

- Arabă
- Chineză (simplificată)
- Chineză (tradițională)
- Coreeană
- Ebraică
- Engleză
- Germană
- Japoneză
- Thailandeză
- Vietnameză
- Chineza, coreeana și japoneza necesită metode speciale de introducere a datelor în formulare, deoarece seturile lor de caractere sunt prea mari pentru a fi introduse în mod practic de la o tastatură, fără tastări multiple.

Limbile Opera

Browserul Opera are un suport excelent pentru limbile europene și pentru afrikaans, care seamănă destul de mult cu olandeza, dar nu se extinde prea mult în afara granițelor Europei continentale. Singurele excepții, desigur, sunt extinderile suportului pentru acele țări atinse de colonialismul european, care a lăsat engleza și alte câteva limbi europene ca limbi vorbite în întreaga lume ca prima sau a doua limbă.

Lista limbilor suportate de Opera:

- Afrikaans
- Engleză
- Franceză
- Germană
- Italiană
- Norvegiană (Bokmaș)
- Norvegiană (Nynorsk)
- Portugheză
- Română
- Spaniolă (europeană)
- Spaniolă (America Latină)
- Suedeză

Alfabete multiple şi alte scrieri

Cascading Style Sheets Level 3 promite să asigure un suport complet pentru codificări multiple ale textului, direcţii de tipărire şi algoritmi de transformare a caracterelor. Aşa cum în engleză se folosesc câteva ligaturi, care reprezintă două caractere combinate într-un singur simbol, cum este ligatura fi care poate fi văzută în tipăriturile de calitate, multe limbi au forme combinate foarte complicate care trebuie să fie prezentate corect pentru lizibilitate. În engleză, aceste combinaţii sunt nişte rafinamente fără care am învăţat să ne descurcăm, deoarece puţine calculatoare pot să le redea, dar nu orice limbă poate fi lipsită de rafinament.

Într-un sens mai larg, XML însuşi va suporta chineza, birmana, scrierea Devanagari, araba, copta şi toate celelalte. HTML a suportat doar limbile vest-europene şi se poate spune că *într-adevăr* era în engleză, deoarece la baza sa era US-ASCII, iar etichetele aveau sens în engleză, cel puţin ca abrevieri, în alte limbi fiind coduri arbitrare.

XML furnizează datele, în timp ce CSS3 şi cele care vor urma asigură formatarea. XSL şi XSLT rămân instrumente specializate, dar probabil vor fi pe server sau vor fi apelate după ce alte procese au hotărât cum vor reprezenta datele în funcţie de schiţe CSS miniaturale gata făcute. În XSL prea multe trebuie create în loc să fie ghicite, nu ca în CSS. Coordonând editările din foile de stil cu reîncărcări periodice ale documentului într-un browser, sau chiar folosind un editor de foi de stil cu un afişaj încorporat, puteţi modifica foaia de stil şi să vedeţi rezultatele în timp real.

Medii multiple

Web-ul devine din ce în ce mai divers iar dispozitivele care îl pot accesa devin din ce în ce mai mici şi mai omniprezente. Deja telefoanele mobile pot citi un e-mail, iar asistenţii digitali personali navighează pe Web. Încep să apară browsere pentru telefoane mobile, dar şi pentru ceasuri de mână. Aceste posibilităţi au apărut datorită câtorva dezvoltări: crearea unor mici limbaje de browser bazate pe XML proiectate special pentru anumite aplicaţii şi dispozitive, suportul CSS2 şi CSS3 pentru diferite tipuri de medii şi transformările XSL/XSLT care convertesc informaţia XML într-o formă acceptabilă pentru fiecare dispozitiv în parte.

Medii interactive

Următoarea generaţie CSS va elimina legăturile dintre afişaj şi redarea statică a unui singur subiect. Foile de stil îşi vor da seama de acţiunile utilizatorului, permiţând astfel creatorilor să elibereze utilizatorii care vor putea să rătăcească în voie pe orice cărări, simplificând producerea de medii de instruire, afişaje pentru educaţie interactivă, manuale de reparaţii şi alte instrumente de învăţare şi cercetare care funcţionează cel mai bine dacă puteţi întoarce pagina pentru o altă vedere.

În combinaţie cu răspunsul vocal, vom fi capabili să controlăm prin vorbire un afişaj rapid cu scheme mobile şi liste de piese. „Nu, nu asta. Următoarea. Asta e! Trimite comanda." Cu o reţea fără cabluri şi un afişaj încorporat în lentilele de contact, aţi termina rapid cu competiţia de la *Rişti şi câştigi*.

Sau, imaginaţi-vă un articol fără legături hipermedia în stil vechi, care vă permite să derivaţi de la un concept în mod dinamic, pur şi simplu indicând un cuvânt sau o frază

şi cerând mai multe informaţii cu un singur gest. Încercuind literele „IBM" vă veţi conecta la pagina lor de bază, iar subliniind literele veţi afla valoarea actuală a acţiunilor şi ultimele noutăţi. Mediile într-adevăr interactive vor avea grijă singure de legături şi vor decide, din experienţa lor îndelungată, tipul informaţiilor pe care le doriţi.

Însă între timp, producătorii de bază, Netscape şi Microsoft, încearcă să ne apropie de acest viitor cu soluţii provizorii care sunt încă, din nefericire, patentate.

Sistemul de acţiune în cascadă Netscape

Script-urile au devenit iritante în multe pagini, fiind nevoie de o durată de încărcare prea mare înainte de a se întâmpla ceva. Netscape a creat un sistem numit Cascading Action Systems (sisteme de acţiune în cascadă), care promite să debaraseze clientul de o parte din timpul de încărcare şi să plaseze comportamentele în pachete refolosibile care pot fi plasate în cache-ul clientului sau chiar în cel al serverului. Desigur, se joacă singuri cu aceste concepte, deoarece Microsoft are o schemă similară, pe care a numit-o Scriptlets, şi comportamente care împachetează HTML şi XML pentru refolosire.

Comportamentele DHTML Microsoft

Mai ţineţi minte Dynamic HTML? Acest aşa-zis standard a declanşat războiul dintre Netscape şi Microsoft şi a încetinit ritmul inovaţiilor, crescând în acelaşi timp costurile de dezvoltare pentru ani buni. Acum a reapărut ca o extindere a CSS care vă permite să construiţi etichete personalizate ce folosesc script-uri pentru a defini acţiuni şi evenimente ale paginii. Un comportament poate stabili poziţia unui element pe ecran, de exemplu, astfel încât să se legene într-o parte şi alta când utilizatorul mută mouse-ul sub el. Obiectul poate fi un şoarece care se plimbă pe ecran după cursor. Etichetele de comportament pot servi ca şi componente refolosibile care încapsulează funcţionalităţi şi comportamente pe o pagină Web. Ca extindere a CSS, comportamentele DHTML combină atribute de acţiune cu atribute de stil în acelaşi cod. Asta înseamnă că foile de stil Microsoft sunt cu atât mai departe de respectarea oricărui standard, cu excepţia standardului lor de exces nestăpânit.

Secţiunea următoare vă arată cum puteţi realiza unele din aceste lucruri într-un mod controlat şi standardizat care este în concordanţă cu scopurile XML.

VRML (Virtual Reality Modeling) – modelarea realităţii virtuale şi XML

VRML, limbajul de modelare a realităţii virtuale (Virtual Reality Modeling Language), este un standard ISO care permite modelarea şi manipularea unor suprafeţe tridimensionale folosind o interfaţă standard. Combinaţia dintre limbajul chimic de marcare CML, bazat pe XML, şi VRML dă posibilitatea cercetătorilor să genereze automat reprezentări VRML ale obiectelor CML într-un mod foarte adevărat pentru înţelegerea structurii tridimensionale a compuşilor şi moleculelor din baza de date CML. După aceasta, un agent inteligent ar putea să „exploateze" baza de cunoştinţe existente căutând moduri noi şi interesante în care moleculele s-ar putea combina.

Ceea ce înseamnă că, la rândul său, un doctor care caută un medicament pentru cancer poate să descrie fragmente importante din proteinele tumorii şi să pornească un agent care să caute interacţiunile potenţiale cu alte molecule. Analog, după ce au fost găsiţi candidaţi, animaţii 3D similare vor permite cercetătorului să manipuleze ambele mole-

cule pentru a căuta porțiuni în care se pot combina.

Deși încă nu *știm* totul, ne apropiem rapid de momentul în care toate datele ne stau la dispoziție și tot ce va mai trebui să facem va fi să le studiem în toate domeniile. Genomul uman complet (ADN-ul care ne definește din punct de vedere genetic) este în curs de descriere. Ce vom face cu acest univers de informații depinde în mare măsură de instrumentele pe care le vom putea îndrepta către el. Lăsând metafizica la o parte, înțelegerea celor aproape trei miliarde de baze ale genomului va pune la grea încercare capacitățile existente. Pentru început vom avea nevoie de o mai bună integrare și interoperabilitate între bazele de date, XML fiind capabil să asigure așa ceva, și de comunicare între cercetătorii și specialiștii în medicină, care este deja facilitată de XML/CML.

În momentul de față, se bănuiește că mult mai multe boli decât putem dovedi efectiv au o bază genetică. Utilizarea unor tehnici VRML rapide pentru a căuta prin întregul genom după structuri specifice poate îmbunătăți în mod spectaculos abilitatea noastră de a prezice și trata bolile. Având unele cunoștințe despre gena susceptibilă, pot fi create tratamente genetice care să repare acea bucățică și să elimine astfel boala din rădăcină, înainte ca persoana să fie efectiv bolnavă.

Microsoft Chrome

Chrome este următoarea mișcare a Microsoft în direcția publicării pe Web a tehnologiilor lor patentate, ActiveX, DNA, ASP, Exchange, OLE DB și DHTML IE4. Chrome permite autorilor HTML și XML să acceseze DirectX din HTML, aceștia putând să integreze direct conținutul multimedia, grafica de înaltă fidelitate, fluxurile media și accelerarea hardware în paginile Web pentru un nivel și mai mare de îndrăzneală pe desktop. Cerințele actuale minime pentru sistem sunt un procesor Pentium II la 350 MHz, un bus de 100 MHz, 4 MB memorie video, 64 MB RAM și capacități DVD, dar, dacă ne luăm după istoria lor, probabil veți avea nevoie de două ori mai mult din fiecare ca să aveți un sistem funcțional. Din fericire, va fi din ce în ce mai ieftin.

Pe de altă parte, utilizarea Chrome vă duce pe partea întunecată a Web-ului. Efectele 3D fanteziste legate de un singur sistem de operare și create în special pentru a distra o audiență pasivă nu-i ajută prea mult pe cei care vor să lucreze. Deși, teoretic, Chrome ar putea să fie folosit și în scopuri utile, pare să fie creat în special pentru a mări atractivitatea și strălucirea reclamelor, succesul jocurilor și interactivitatea conținutului DVD și multimedia.

XML în sine se mișcă spre combinarea mai multor tehnologii și fluxuri de date în documente create, în parte, în funcție de o combinație de cerințe ale utilizatorului și ale dispozitivului. Secțiunea următoare vă descrie cum pot fi îmbinate în documente XML și afișate într-o singură fereastră fluxurile de date multiple, care în momentul de față necesită rularea mai multor aplicații pentru redarea lor pe ecran.

Fluxuri de date multiple

Chiar în momentul de față putem avea un formular de știri sau un broker de acțiuni care să funcționeze continuu într-o fereastră, în timp ce într-o altă fereastră mică urmărim fără prea multă atenție o emisiune sportivă televizată, în timp ce trimitem un e-mail cu o mână, iar cu cealaltă completăm cifrele de vânzare. Din nefericire, toate acestea se petrec în mai multe ferestre. Ar fi și mai bine dacă toate ar putea intra în aceeași fereastră. Nu există nici un motiv temeinic pentru care fiecare aplicație cere o fereastră separată, în afară de obișnuințele creatorilor de aplicații monolitice care nu văd nici un motiv să cedeze o parte din spațiul de pe ecran pentru nevoile altor aplicații.

Documentele XML, care prin design pot fi aproape orice dorim – de la Dialogurile lui Platon la repriza a doua a derby-ului campionatului – pot fi cu ușurință contopite într-un singur document. Aceste documente pot avea aproape orice înfățișare dorim. Deci am putea, teoretic, să creăm un document care să accepte cifrele de vânzări ca intrări și, printr-o deplasare ușoară a focalizării, să completăm un e-mail în același timp și toate acestea fără a pierde din ochi scorul. Iar toate se întâmplă pe aceeași pagină.

Abilitatea de a uni și manipula fluxuri de date multiple depășește, evident, acest exemplu banal, deoarece o abilitate similară ar putea permite aceluiași ecran să joace rol de ecran de introducere a comenzilor, de afișaj de control al inventarului și de platformă video interactivă. Această abilitate este o teroare pentru creatorii de sisteme de operare, deoarece un browser XML creat să funcționeze pe orice platformă hardware poate să creeze un aspect și o atmosferă uniformă și o interfață cu utilizatorul unitară care poate să ascundă față de utilizator serviciile sistemului, sau chiar să le ocolească complet.

Ceea ce este rău pentru producătorii de sisteme de operare este foarte bine pentru creatorii de aplicații, deoarece aceștia vor putea să scrie un cod pentru o interfață uniformă de browser XML, iar acesta va rula pe orice platformă din lume fără înregistrări sau muncă suplimentară de dezvoltare.

În practica actuală, acest lucru s-ar putea face prin definirea de notații pentru fiecare formă de intrare și plasarea ulterioară a documentului pe o pagină cu CSS sau XSL. În viitor, poate se va putea face același lucru printr-un mecanism mai general, care va presupune că dumneavoastră indicați tipul de intrare pe care îl doriți, iar serverul va decide printr-o interacțiune cu browserul care flux de date se va potrivi mai bine pentru dumneavoastră.

În serviciul mai multor tipuri de utilizatori

Web-ul viitorului va cuprinde o secțiune largă a populației, în care va fi reprezentată fiecare categorie de persoane, vorbind limbi diferite, având abilități diferite și concepții diferite despre ceea ce așteaptă de la Internet.

Dispozitivele vor fi legate la Web, un tehnician virtual putând să monitorizeze în timp temperatura din frigiderul dumneavoastră și să vă anunțe dacă izolația ușii este defectă și trebuie înlocuită. Chiar la timp, deoarece în industrie au apărut sisteme de control al inventarului. Un sistem similar ar putea să examineze ce intră și ce iese din bucătăria dumneavoastră și să acceseze o bază de date personalizată pentru a comanda mărfuri automat. Felurile de mâncare mai speciale pot fi comandate după nume, iar „procesorul de hrană“ (un termen vechi cu un înțeles nou) poate să genereze o listă de ingrediente și să anunțe la piață ce produse sunt necesare pentru masă și când ar trebui să sosească.

Chiar dacă mersul la cumpărături este un mod grozav de a vă întâlni cu alții, majoritatea persoanelor au lucruri mai importante cu care pot să își ocupe timpul. Pentru mulți, timpul este mai scump decât cumpărăturile. O piață fără cumpărători fizici în interior poate să întreacă o hală pentru comerț en-gros. Se pot elimina spațiul culoarelor, casele

de marcare și o cauză importantă a alterării alimentelor și a risipei: cumpărătorii care nu se pot abține și gustă produsele.

Viitorul HTML

Chiar pe termen lung, este greu de crezut că HTML va dispărea. Este prea comod și se adresează unei audiențe prea mari. La fel cum televiziunea nu a eliminat radioul, nici XML nu va elimina HTML. Totuși, ce se *va* întâmpla este că HTML va deveni mai mult un produs de nișă, și nu soluția universală în care am încercat să îl transformăm astăzi. Numărul limbajelor de afișare va crește o dată cu răspândirea dispozitivelor specializate, iar instrumentele care vor manipula diferite limbaje de marcare vor fi omniprezente.

Deci vom avea cu toții experiențe diferite cu aceleași date esențiale. Dacă sunteți conectat la consola dumneavoastră de realitate virtuală, veți avea o experiență cu totul diferită față de mine, care voi accesa același sit de pe telefonul meu mobil. Dar rolurile și punctele noastre de vedere se vor schimba în fiecare zi. Consola vă rămâne acasă atunci când dumneavoastră vă aflați la volanul mașinii, trecând prin Web și prin bazele sale de date la fel ca vecinul cu handicap de vedere, iar când ajungeți la serviciu, unde aveți o conexiune la rețea de un gigabit, totul se schimbă din nou.

Ce se va schimba în mod sigur este asaltul actual asupra timpului nostru din partea proiectanților care realizează un sit care este grozav când stă pe desktop-ul lor, iar apoi îl lipesc pe un server antic, ascuns undeva într-un dulap, și care este conectat la Internet printr-un pai. Nu pot să concep să mă întorc la un modem telefonic acum, după ce am cunoscut experiența DSL, și nu vom putea concepe să ne întoarcem la situația de acum după ce vom ști ce poate face într-adevăr XML.

Alternativele viitorului

Lucrurile nu au ieșit așa cum speram

După o planificare minuțioasă, pregătirea câtorva rapoarte, a unui plan de afaceri, a unui prospect, a unei oferte de acțiuni sau a unei cereri în căsătorie, lucrurile nu au ieșit așa cum ne așteptam.

Autostrada este o bună metaforă pentru procesul de planificare. Bazându-vă pe o hartă rutieră, puteți să planificați o rută cu mult înainte de vacanța în Grand Canyon. În funcție de informațiile zilnice despre autostrăzi furnizate de poliția autostrăzii sau de biroul poliției statale, puteți face modificări pe termen mediu ale planurilor dumneavoastră înainte de a porni în călătorie. Rapoartele radio vă pot ajuta să actualizați planurile chiar pe drum. Însă informația cea mai proaspătă este cea pe care o vedeți în fața ochilor, iar această informație poate să implice evitări sau frâne bruște.

Unele evenimente neașteptate necesită doar o visare tihnită sau un grup focus din când în când, fără nici o grabă, dacă nu afectează planurile actuale. Dar alte surprize cer un răspuns imediat. Un bun planificator, ca un șofer bun, nu pierde niciodată din vedere scopul propus, dar nu întârzie să răspundă în siguranță la condițiile care se modifică.

Pentru XML, ca de altfel pentru orice alt efort uman, recompensa va fi culeasă de cei mai rapizi, cei mai norocoși și cei mai inteligenți, în această ordine. Observați că rapiditatea și norocul bat inteligența în această listă. Dacă reușiți să le aveți pe toate trei, veți fi într-o poziție privilegiată.

În momentul de față, XML este limitat prin mecanismul notațiilor doar la mecanismele multimedia care pot fi imaginate astăzi. Dacă tehnologia avansează, cum a făcut-o până acum, pentru a profita de ea este nevoie de un nou document. Dar acest lucru poate să se schimbe. Dacă acum avem uneori de trecut prin chinul actualizării unor situri întregi pentru ca schimbarea să se producă, poate că în curând va fi posibil ca extinderea să fie într-o oarecare măsură compatibilă cu prezentul.

Poate că în curând va fi posibil să înlocuim fără piedici imagini GIF cu PNG-uri, iar apoi pe acestea cu imagini holografice tridimensionale sau chiar cu ferestre virtuale spre obiecte existente în timp real. Deja vedem începuturile primitive ale acestui tip de tehnologie în camerele Web de luat vederi și în Jocurile 3-D. Mai avem de așteptat doar noile invenții ale imaginației umane și noile produse create din părți ale invențiilor mai vechi plămădite împreună cu ideile cele mai noi.

Este interesant, gândindu-ne la XML, să comparăm viziunile asupra viitorului.

Pe de o parte, avem lumi viitoare ca cea din *Star Trek* și descendenții săi, în care tehnologia este foarte vizibilă. În *Star Trek*, chiar de la început existau calculatoare care vorbesc, cărți electronice și mașini foarte complexe a căror funcționalitate și ale căror principii de funcționare sunt cunoscute până și de copii și sunt în serviciul tuturor vieţuitoarelor. Această lume este caracterizată de o distincție clară între mașini și oameni, și de perspective umane pline de speranță. Chiar și mașinile inteligente din această lume aspiră, ca și Pinocchio, la o calitate umană. XML pare să se potrivească bine cu această lume, deoarece are ca scop fundamental satisfacerea directă a nevoilor umane. XML pare potrivit pentru a implementa un calculator vorbitor sau o carte electronică, și chiar manipularea unor mașini complexe într-un limbaj înțeles și de copii poate să devină posibilă prin canalele de comunicare XML.

Pe de altă parte, avem lumile mai inumane, cele din *Bladerunner* (*Vânătorul de recompense*), din viziunile cyberpunk, poate chiar cele din *Dune* sau *Star Wars* (*Războiul stelelor*). În acestea, tehnologia este misterioasă, aproape mută, invizibilă sau antropomorfizată până la punctul în care mașinile se plimbă și vorbesc ca oamenii, dar nu aspiră la acest statut, iar oamenii au implanturi mecanice, astfel încât granița dintre oameni și mașini este neclară. În aceste viziuni, oamenii de succes aspiră să devină mașini, sau sunt în secret mașini, iar omenirea este condamnată. Cum XML este doar o tehnologie și, în acest sens, este neutru, orice scenariu pare posibil, depinzând de concepția dumneavoastră, mai mult sau mai puțin optimistă, dar XML pare democratic într-un sens mai profund, și poate mai potrivit pentru prima variantă decât pentru a doua. Dar nu se știe niciodată.

În acest moment noi vedem doar începutul posibilităților. Ce se va întâmpla în realitate depinde de dumneavoastră.

Lucrurile au ieșit exact așa cum am prezis

După ce ați cumpărat biletul de loterie la sfatul unei țigănci vrăjitoare și ați câștigat imediat premiul cel mare, ați crezut că nu se poate mai bine de atât. Dar iată că se poate!

Bun, deci acum sunteți mai bogat decât în visele dumneavoastră cele mai lacome. Este o perioadă periculoasă. Mai mulți decât credeți au fost ruinați în plină glorie. Gândiți-vă

în continuare la drumul pe care îl aveţi de parcurs. Poate vine din faţă un camion.

Deşi succesul XML ca întreg pare asigurat, există multe probleme care au importanţă nu numai din punct de vedere tehnic. Legislaţia poate să influenţeze comerţul electronic în general prin taxe pe venit sau pe vânzări impuse în diferite jurisdicţii din lumea întreagă. Preocupările faţă de confidenţialitate pot să intre în prim-plan, segmente întregi din domeniu fiind forţate să îşi restructureze fluxurile de venituri dacă SUA şi Uniunea Europeană, în particular, nu reuşesc să dea formă unui compromis între inerţia SUA în ceea ce priveşte confidenţialitatea şi preocuparea europeană în domeniu. Poate vom înclina spre modelul SUA sau ne vom îndrepta spre modelul de confidenţialitate european la scara întregii lumi. Modelul învingător va schimba complet lucrurile.

În acest moment, există în fiecare ţară forţe importante care doresc să se păstreze această stare de fapt. Acestea poate că se vor impune pe termen scurt. Însă în piaţa globală, în care SUA nu mai este o forţă la fel de proeminentă ca în trecut, dar mai are încă un cuvânt greu de spus, chiar şi marile puteri vor prefera un compromis şi nu o hegemonie. În final, poate vom ajunge la o piaţă mondială în care toată lumea va respecta mai mult sau mai puţin aceleaşi reguli. Dar ale cui reguli vor prevala nu este încă foarte clar.

Politicile interne vor interveni şi ele, dacă propriul electorat va decide să legifereze unele restricţii pentru Internet. O mare parte din Web este finanţată din veniturile din reclame, valoarea totală a acestora depinzând de cât de multă informaţie de identificare este furnizată la fiecare interacţiune a utilizatorului cu serverul. Dacă o legislaţie restrictivă coboară aceste valori, poate chiar compania dumneavoastră va fi lipsită de orice libertate de acţiune sau va fi pregătită pentru a profita de noile ocazii de afaceri. Care vor fi aceste ocazii nu poate nimeni să ştie exact, dar cei cu imaginaţia cea mai activă şi curiozitatea cea mai viguroasă vor afla primii. Creativitatea nu doarme niciodată.

Referinţe XML/XHTML

ANEXA

A

În această anexă

Referinţa tehnică XML
Referinţa tehnică XHTML

Referinţa tehnică XML

XML 1.0 este descris mai mult sau mai puţin în Recomandarea W3C XML 1.0 prin cele 89 de producţii ale sale listate în forma extinsă Backus-Naur (EBNF – Extended Backus-Naur Form), cu câteva avertismente numite constrângeri de validitate şi constrângeri de construcţie corectă, şi unul sau două atribute încorporate pentru echilibrare. Însă forma Backus-Naur nu este foarte transparentă pentru cei care nu sunt obişnuiţi să extragă informaţii utile din descrieri prescurtate obscure. De fapt, multe din instrumentele existente, scrise de programatori adevăraţi, nu reuşesc să execute complet şi corect specificaţia, fie din cauza presiunilor de marketing care au grăbit lansarea produsului, fie din cauza greşelilor de proiectare. Pentru a înţelege complet Recomandarea, cineva ar trebui să înţeleagă o sintaxă aproape la fel de complexă ca şi XML 1.0 în sine şi de două ori mai obscură, deci este destul de complicat. În fine, o mare parte a Recomandării a fost exprimată de W3C în cuvintele care însoţesc EBNF, atât de îndrăgită de programatori. Documentul trebuie într-adevăr să fie citit în întregime, cu grija cu care un preot citeşte Biblia, pentru ca profunzimea sa să apară din obscuritate.

În această anexă sunt prezentate regulile Backus-Naur şi sunt explicate foarte simplu, apoi apar câteva exemple care să vă dea o idee despre înţelesul acestor reguli. Trebuie să fie folosită ca referinţă scurtă, care completează discuţiile mai lungi despre DTD-uri din celelalte capitole ale cărţii. În orice caz, vă sfătuiesc să studiaţi specificaţia însăşi şi să nu vă bazaţi pe explicaţii decât ca fituici care ilustrează sau extind cuvintele din specificaţie. Întreaga specificaţie oficială poate fi găsită la `http://www.w3.org/TR/REC-x`.

În mod surprinzător, sau poate nu chiar, dacă vă gândiţi că simplitatea a fost un scop major al dezvoltatorilor, XML poate fi definit prin numai 89 de producţii EBNF şi câteva citiri atente ale textului recomandării. Producţiile descriu descompunerea fiecărei variabile. Când ajungeţi la producţii care sunt încadrate în semne de citat, aţi ajuns la un nod terminal şi poate chiar la un cuvânt cheie sau la o constantă.

Listingul A.1 vă prezintă toate producţiile EBNF descrise în Recomandarea XML 1.0. Notaţiile folosite în producţiile `[13] ([-'()+,./:=?;!*#@$_%])` şi `[26] ([a-zA-Z0-9_.:])` nu sunt descrise precis, deşi sunt de fapt identice cu cele folosite în expresiile regulate

din UNIX (Unix-style Regular Expressions), o listă enumerată alcătuită dintr-o mulțime de caractere încadrate de paranteze drepte. Notația cu cratimă din producția 26 este de fapt o prescurtare a enumerării unui interval echivalent cu „abcdefgh ijklmnopqrstuvwxyz" și „ABCDEFGHIJKLMNOPQRSTUVWXYZ" și, respectiv, „0123456789", deși ordinea alfabetică și convențiile de interval presupuse aici nu se traduc foarte bine în lumea IBM EBCDIC. Observați de asemenea că prima cratimă din producția 13 nu este ambiguă, deoarece este chiar primul caracter din listă.
În listing, constrângerile de validitate și de construcție corectă sunt identificate prin paranteze drepte, într-o formă mnemonică concisă. Pentru explicații complete, vezi Capitolele 2 și 3 sau chiar Recomandarea XML 1.0.

▶ **Vezi** „Constrângerile construirii corecte", la **pagina 41.**

▶ **Vezi** „Diferența dintre un XML construit corect și un XML valid", la **pagina 83.**

În Listingul A.1, constrângerile de validitate și de construcție corectă sunt prezentate astfel:

```
[ VC: Validity Constrain ]
[ WFC: Well-Formedness Constrain ]
```

VC înseamnă Validity Constrain (constrângere de validitate) și este urmată de descrierea concisă a constrângerii. WFC înseamnă Well-Formedness Constrain (constrângere de construcție corectă) și, similar, este urmată de descrierea concisă a constrângerii. Acestea sunt doar reguli mnemonice pentru a reține constrângerea. Înainte de a trage concluzii, trebuie să citiți textul care descrie constrângerea și uneori chiar și textul însoțitor.

Listingul A.1 Gramatica XML în forma Backus-Naur extinsă (EBNF)

```
[1]  document          ::= prolog element Misc*
[2]  Char              ::= #x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD]
                         | [#x10000-#x10FFFF]
   /* any Unicode character, excluding the surrogate blocks, FFFE, and FFFF. */
[3]  S                 ::= (#x20 | #x9 | #xD | #xA)+
                           /* Whitespace is space, tab, cr, lf */
[4]  NameChar          ::= Letter | Digit | '.' | '-' | '_' | ':'
                         | CombiningChar | Extender
[5]  Name              ::= (Letter | '_' | ':') (NameChar)*
[6]  Names             ::= Name (#x20 Name)*
[7]  Nmtoken           ::= (NameChar)+
[8]  Nmtokens          ::= Nmtoken (#x20 Nmtoken)*
[9]  EntityValue       ::= '"' ([^%&"] | PEReference | Reference)* '"'
                         | "'" ([^%&'] | PEReference | Reference)* "'"
[10] AttValue          ::= '"' ([^<&"] | Reference)* '"' | "'" ([^<&']
                         | Reference)* "'"
[11] SystemLiteral     ::= ('"' [^"]* '"') | ("'" [^']* "'")
[12] PubidLiteral      ::= '"' PubidChar* '"' | "'" (PubidChar - "'")* "'"
[13] PubidChar         ::= #x20 | #xD | #xA | [a-zA-Z0-9]
                         | [-'()+,./:=?;!*#@$_%]
[14] CharData          ::= [^<&]* - ([^<&]* ']]>' [^<&]*)
[15] Comment           ::= '<!--' ((Char - '-') | ('-' (Char - '-')))* '-->'
[16] PI                ::= '<?' PITarget (S (Char* - (Char* '?>' Char*)))? '?>'
[17] PITarget          ::= Name - (('X' | 'x') ('M' | 'm') ('L' | 'l'))
[18] CDSect            ::= CDStart CData CDEnd
[19] CDStart           ::= '<![CDATA['
[20] CData             ::= (Char* - (Char* ']]>' Char*))
[21] CDEnd             ::= ']]>'
[22] prolog            ::= XMLDecl? Misc* (doctypedecl Misc*)?
[23] XMLDecl           ::= '<?xml' VersionInfo EncodingDecl? SDDecl? S? '?>'
[24] VersionInfo       ::= S 'version' Eq ("'" VersionNum "'" | '"' VersionNum
                         '"')
[25] Eq                ::= S? '=' S?
[26] VersionNum        ::= ([a-zA-Z0-9_.:] | '-')+
[27] Misc              ::= Comment | PI | S
[28] doctypedecl       ::= '<!DOCTYPE' S Name (S ExternalID)?
                           S? ('[' (markupdecl | PEReference | S)* ']' S?)? '>'
                                  [ VC: Root Element Type ]
[29] markupdecl        ::= elementdecl | AttlistDecl | EntityDecl
                         | NotationDecl | PI | Comment
                                  [ VC: Proper Declaration/PE Nesting ]
                                  [ WFC: PEs in Internal Subset ]
[30] extSubset         ::= TextDecl? extSubsetDecl
[31] extSubsetDecl     ::= ( markupdecl | conditionalSect | PEReference | S )*
[32] SDDecl            ::= S 'standalone' Eq (("'" ('yes' | 'no') "'")
                         | ('"' ('yes' | 'no') '"'))
                                  [ VC: Standalone Document Declaration ]
[33] LanguageID        ::= Langcode ('-' Subcode)*
[34] Langcode          ::= ISO639Code | IanaCode | UserCode
[35] ISO639Code        ::= ([a-z] | [A-Z]) ([a-z] | [A-Z])
[36] IanaCode          ::= ('i' | 'I') '-' ([a-z] | [A-Z])+
[37] UserCode          ::= ('x' | 'X') '-' ([a-z] | [A-Z])+
[38] Subcode           ::= ([a-z] | [A-Z])+
[39] element           ::= EmptyElemTag | STag content ETag
                                  [ WFC: Element Type Match ]
                                  [ VC: Element Valid ]
[40] STag              ::= '<' Name (S Attribute)* S? '>'
                                  [ WFC: Unique Att Spec ]
[41] Attribute         ::= Name Eq AttValue
                                  [ VC: Attribute Value Type ]
                                  [ VC: Valid xml:lang ]
                                  [ WFC: No External Entity References ]
                                  [ WFC: No < in Attribute Values ]
[42] Etag              ::= '</' Name S? '>'
[43] content           ::= (element | CharData | Reference | CDSect
                         | PI | Comment)*
[44] EmptyElemTag      ::= '<' Name (S Attribute)* S? '/>'
                                  [ WFC: Unique Att Spec ]
[45] elementdecl       ::= '<!ELEMENT' S Name S contentspec S? '>'
                                  [ VC: Unique Element Type Declaration ]
[46] contentspec       ::= 'EMPTY' | 'ANY' | Mixed | children
[47] children          ::= (choice | seq) ('?' | '*' | '+')?
[48] cp                ::= (Name | choice | seq) ('?' | '*' | '+')?
[49] choice            ::= '(' S? cp ( S? '|' S? cp )+ S? ')'
                                  [ VC: Proper Group/PE Nesting ]
[50] seq               ::= '(' S? cp ( S? ',' S? cp )* S? ')'
                                  [ VC: Proper Group/PE Nesting ]
[51] Mixed             ::= '(' S? '#PCDATA' (S? '|' S? Name)* S? ')*'
```

Listingul A.1 (continuare)

```
                    | '(' S? '#PCDATA' S? ')'
                                [ VC: Proper Group/PE Nesting ]
                                [ VC: No Duplicate Types ]
[52] AttlistDecl    ::= '<!ATTLIST' S Name AttDef* S? '>'
[53] AttDef         ::= S Name S AttType S DefaultDecl
[54] AttType        ::= StringType | TokenizedType
                      | EnumeratedType
[55] StringType     ::= 'CDATA'
[56] TokenizedType  ::= 'ID'
                                [ VC: ID ]
                                [ VC: One ID per Element Type ]
                                [ VC: ID Attribute Default ]
                      | 'IDREF'
                                [ VC: IDREF ]
                      | 'IDREFS'
                                [ VC: IDREF ]
                      | 'ENTITY'
                                [ VC: Entity Name ]
                      | 'ENTITIES'
                                [ VC: Entity Name ]
                      | 'NMTOKEN'
                                [ VC: Name Token ]
                      | 'NMTOKENS'
                                [ VC: Name Token ]
[57] EnumeratedType ::= NotationType | Enumeration
[58] NotationType   ::= 'NOTATION' S '(' S? Name (S? '|' S? Name)* S? ')'
                                [ VC: Notation Attributes ]
                                [ VC: One Notation per Element Type ]
[59] Enumeration    ::= '(' S? Nmtoken (S? '|' S? Nmtoken)* S? ')'
                                [ VC: Enumeration ]
[60] DefaultDecl    ::= '#REQUIRED' | '#IMPLIED' | (('#FIXED' S)? AttValue)
                                [ VC: Required Attribute ]
                                [ VC: Attribute Default Legal ]
                                [ WFC: No < in Attribute Values ]
                                [ VC: Fixed Attribute Default ]
[61] conditionalSect ::= includeSect | ignoreSect
[62] includeSect    ::= '<![' S? 'INCLUDE' S? '[' extSubsetDecl ']]>'
[63] ignoreSect     ::= '<![' S? 'IGNORE' S? '[' ignoreSectContents* ']]>'
[64] ignoreSectContents ::= Ignore ('<![' ignoreSectContents ']]>' Ignore)*
[65] Ignore         ::= Char* - (Char* ('<![' | ']]>') Char*)
[66] CharRef        ::= '&#' [0-9]+ ';' | '&#x' [0-9a-fA-F]+ ';'
                                [ WFC: Legal Character ]
[67] Reference      ::= EntityRef | CharRef
[68] EntityRef      ::= '&' Name ';'
                                [ WFC: Entity Declared ]
                                [ VC: Entity Declared ]
                                [ WFC: Parsed Entity ]
                                [ WFC: No Recursion ]
[69] PEReference    ::= '%' Name ';'
                                [ VC: Entity Declared ]
                                [ WFC: No Recursion ]
                                [ WFC: In DTD ]
```

```
[70] EntityDecl     ::= GEDecl | PEDecl
[71] GEDecl         ::= '<!ENTITY' S Name S EntityDef S? '>'
[72] PEDecl         ::= '<!ENTITY' S '%' S Name S PEDef S? '>'
[73] EntityDef      ::= EntityValue | (ExternalID NDataDecl?)
[74] PEDef          ::= EntityValue | ExternalID
[75] ExternalID     ::= 'SYSTEM' S SystemLiteral
                      | 'PUBLIC' S PubidLiteral S SystemLiteral
[76] NdataDecl      ::= S 'NDATA' S Name [ VC: Notation Declared ]
[77] TextDecl       ::= '<?xml' VersionInfo? EncodingDecl S? '?>'
[78] extParsedEnt   ::= TextDecl? content
[79] extPE          ::= TextDecl? extSubsetDecl
[80] EncodingDecl   ::= S 'encoding' Eq ('"' EncName '"'
                      | "'" EncName "'" )
[81] EncName        ::= [A-Za-z] ([A-Za-z0-9._] | '-')*
                      /* Encoding name contains only Latin characters */
[82] NotationDecl   ::= '<!NOTATION' S Name S (ExternalID
                      | PublicID) S? '>'
[83] PublicID       ::= 'PUBLIC' S PubidLiteral
[84] Letter         ::= BaseChar | Ideographic
[85] BaseChar       ::= [#x0041-#x005A] | [#x0061-#x007A] | [#x00C0-#x00D6]
                      | [#x00D8-#x00F6]
                      | [#x00F8-#x00FF] | [#x0100-#x0131] | [#x0134-#x013E]
                      | [#x0141-#x0148] | [#x014A-#x017E] | [#x0180-#x01C3]
                      | [#x01CD-#x01F0] | [#x01F4-#x01F5] | [#x01FA-#x0217]
                      | [#x0250-#x02A8] | [#x02BB-#x02C1] | #x0386
                      | [#x0388-#x038A]
                      | #x038C | [#x038E-#x03A1] | [#x03A3-#x03CE]
                      | [#x03D0-#x03D6]
                      | #x03DA | #x03DC | #x03DE | #x03E0 | [#x03E2-#x03F3]
                      | [#x0401-#x040C] | [#x040E-#x044F] | [#x0451-#x045C]
                      | [#x045E-#x0481] | [#x0490-#x04C4] | [#x04C7-#x04C8]
                      | [#x04CB-#x04CC] | [#x04D0-#x04EB] | [#x04EE-#x04F5]
                      | [#x04F8-#x04F9] | [#x0531-#x0556] | #x0559
                      | [#x0561-#x0586]
                      | [#x05D0-#x05EA] | [#x05F0-#x05F2] | [#x0621-#x063A]
                      | [#x0641-#x064A] | [#x0671-#x06B7] | [#x06BA-#x06BE]
                      | [#x06C0-#x06CE] | [#x06D0-#x06D3] | #x06D5
                      | [#x06E5-#x06E6]
                      | [#x0905-#x0939] | #x093D | [#x0958-#x0961]
                      | [#x0985-#x098C]
                      | [#x098F-#x0990] | [#x0993-#x09A8] | [#x09AA-#x09B0]
                      | #x09B2
                      | [#x09B6-#x09B9] | [#x09DC-#x09DD] | [#x09DF-#x09E1]
                      | [#x09F0-#x09F1] | [#x0A05-#x0A0A] | [#x0A0F-#x0A10]
                      | [#x0A13-#x0A28] | [#x0A2A-#x0A30] | [#x0A32-#x0A33]
                      | [#x0A35-#x0A36] | [#x0A38-#x0A39] | [#x0A59-#x0A5C]
                      | #x0A5E
                      | [#x0A72-#x0A74] | [#x0A85-#x0A8B] | #x0A8D
                      | [#x0A8F-#x0A91]
                      | [#x0A93-#x0AA8] | [#x0AAA-#x0AB0] | [#x0AB2-#x0AB3]
                      | [#x0AB5-#x0AB9] | #x0ABD | #x0AE0 | [#x0B05-#x0B0C]
                      | [#x0B0F-#x0B10] | [#x0B13-#x0B28] | [#x0B2A-#x0B30]
                      | [#x0B32-#x0B33] | [#x0B36-#x0B39] | #x0B3D
```

Listingul A.1 (continuare)

```
                    | [#x0B5C-#x0B5D]
                    | [#x0B5F-#x0B61] | [#x0B85-#x0B8A] | [#x0B8E-#x0B90]
                    | [#x0B92-#x0B95] | [#x0B99-#x0B9A] | #x0B9C
                    | [#x0B9E-#x0B9F]
                    | [#x0BA3-#x0BA4] | [#x0BA8-#x0BAA] | [#x0BAE-#x0BB5]
                    | [#x0BB7-#x0BB9] | [#x0C05-#x0C0C] | [#x0C0E-#x0C10]
                    | [#x0C12-#x0C28] | [#x0C2A-#x0C33] | [#x0C35-#x0C39]
                    | [#x0C60-#x0C61] | [#x0C85-#x0C8C] | [#x0C8E-#x0C90]
                    | [#x0C92-#x0CA8] | [#x0CAA-#x0CB3] | [#x0CB5-#x0CB9]
                    | #x0CDE
                    | [#x0CE0-#x0CE1] | [#x0D05-#x0D0C] | [#x0D0E-#x0D10]
                    | [#x0D12-#x0D28] | [#x0D2A-#x0D39] | [#x0D60-#x0D61]
                    | [#x0E01-#x0E2E] | #x0E30 | [#x0E32-#x0E33]
                    | [#x0E40-#x0E45]
                    | [#x0E81-#x0E82] | #x0E84 | [#x0E87-#x0E88]
                    | #x0E8A | #x0E8D
                    | [#x0E94-#x0E97] | [#x0E99-#x0E9F] | [#x0EA1-#x0EA3]
                    | #x0EA5
                    | #x0EA7 | [#x0EAA-#x0EAB] | [#x0EAD-#x0EAE] | #x0EB0
                    | [#x0EB2-#x0EB3] | #x0EBD | [#x0EC0-#x0EC4]
                    | [#x0F40-#x0F47]
                    | [#x0F49-#x0F69] | [#x10A0-#x10C5] | [#x10D0-#x10F6]
                    | #x1100
                    | [#x1102-#x1103] | [#x1105-#x1107] | #x1109
                    | [#x110B-#x110C]
                    | [#x110E-#x1112] | #x113C | #x113E | #x1140 | #x114C
                    | #x114E
                    | #x1150 | [#x1154-#x1155] | #x1159 | [#x115F-#x1161]
                    | #x1163
                    | #x1165 | #x1167 | #x1169 | [#x116D-#x116E]
                    | [#x1172-#x1173]
                    | #x1175 | #x119E | #x11A8 | #x11AB | [#x11AE-#x11AF]
                    | [#x11B7-#x11B8] | #x11BA | [#x11BC-#x11C2] | #x11EB
                    | #x11F0
                    | #x11F9 | [#x1E00-#x1E9B] | [#x1EA0-#x1EF9]
                    | [#x1F00-#x1F15]
                    | [#x1F18-#x1F1D] | [#x1F20-#x1F45] | [#x1F48-#x1F4D]
                    | [#x1F50-#x1F57] | #x1F59 | #x1F5B | #x1F5D
                    | [#x1F5F-#x1F7D]
                    | [#x1F80-#x1FB4] | [#x1FB6-#x1FBC] | #x1FBE
                    | [#x1FC2-#x1FC4]
                    | [#x1FC6-#x1FCC] | [#x1FD0-#x1FD3] | [#x1FD6-#x1FDB]
                    | [#x1FE0-#x1FEC] | [#x1FF2-#x1FF4] | [#x1FF6-#x1FFC]
                    | #x2126
                    | [#x212A-#x212B] | #x212E | [#x2180-#x2182]
                    | [#x3041-#x3094]
                    | [#x30A1-#x30FA] | [#x3105-#x312C] | [#xAC00-#xD7A3]
[86] Ideographic    ::= [#x4E00-#x9FA5] | #x3007 | [#x3021-#x3029]
[87] CombiningChar  ::= [#x0300-#x0345] | [#x0360-#x0361] | [#x0483-#x0486]
                      | [#x0591-#x05A1]
                      | [#x05A3-#x05B9] | [#x05BB-#x05BD] | #x05BF
                      | [#x05C1-#x05C2]
```



```
                      | #x05C4 | [#x064B-#x0652] | #x0670 | [#x06D6-#x06DC]
                      | [#x06DD-#x06DF] | [#x06E0-#x06E4] | [#x06E7-#x06E8]
                      | [#x06EA-#x06ED] | [#x0901-#x0903] | #x093C
                      | [#x093E-#x094C]
                      | #x094D | [#x0951-#x0954] | [#x0962-#x0963]
                      | [#x0981-#x0983]
                      | #x09BC | #x09BE | #x09BF | [#x09C0-#x09C4]
                      | [#x09C7-#x09C8]
                      | [#x09CB-#x09CD] | #x09D7 | [#x09E2-#x09E3] | #x0A02
                      | #x0A3C
                      | #x0A3E | #x0A3F | [#x0A40-#x0A42] | [#x0A47-#x0A48]
                      | [#x0A4B-#x0A4D] | [#x0A70-#x0A71] | [#x0A81-#x0A83]
                      | #x0ABC
                      | [#x0ABE-#x0AC5] | [#x0AC7-#x0AC9] | [#x0ACB-#x0ACD]
                      | [#x0B01-#x0B03] | #x0B3C | [#x0B3E-#x0B43]
                      | [#x0B47-#x0B48]
                      | [#x0B4B-#x0B4D] | [#x0B56-#x0B57] | [#x0B82-#x0B83]
                      | [#x0BBE-#x0BC2] | [#x0BC6-#x0BC8] | [#x0BCA-#x0BCD]
                      | #x0BD7
                      | [#x0C01-#x0C03] | [#x0C3E-#x0C44] | [#x0C46-#x0C48]
                      | [#x0C4A-#x0C4D] | [#x0C55-#x0C56] | [#x0C82-#x0C83]
                      | [#x0CBE-#x0CC4] | [#x0CC6-#x0CC8] | [#x0CCA-#x0CCD]
                      | [#x0CD5-#x0CD6] | [#x0D02-#x0D03] | [#x0D3E-#x0D43]
                      | [#x0D46-#x0D48] | [#x0D4A-#x0D4D] | #x0D57 | #x0E31
                      | [#x0E34-#x0E3A] | [#x0E47-#x0E4E] | #x0EB1
                      | [#x0EB4-#x0EB9]
                      | [#x0EBB-#x0EBC] | [#x0EC8-#x0ECD] | [#x0F18-#x0F19]
                      | #x0F35
                      | #x0F37 | #x0F39 | #x0F3E | #x0F3F | [#x0F71-#x0F84]
                      | [#x0F86-#x0F8B] | [#x0F90-#x0F95] | #x0F97
                      | [#x0F99-#x0FAD]
                      | [#x0FB1-#x0FB7] | #x0FB9 | [#x20D0-#x20DC] | #x20E1
                      | [#x302A-#x302F] | #x3099 | #x309A
[88] Digit          ::= [#x0030-#x0039] | [#x0660-#x0669] | [#x06F0-#x06F9]
                      | [#x0966-#x096F]
                      | [#x09E6-#x09EF] | [#x0A66-#x0A6F] | [#x0AE6-#x0AEF]
                      | [#x0B66-#x0B6F] | [#x0BE7-#x0BEF] | [#x0C66-#x0C6F]
                      | [#x0CE6-#x0CEF] | [#x0D66-#x0D6F] | [#x0E50-#x0E59]
                      | [#x0ED0-#x0ED9] | [#x0F20-#x0F29]
[89] Extender       ::= #x00B7 | #x02D0 | #x02D1 | #x0387 | #x0640 | #x0E46
                      | #x0EC6 | #x3005
                      | [#x3031-#x3035] | [#x309D-#x309E] | [#x30FC-#x30FE]
```

Note la fişuica XML

Când încercaţi să înţelegeţi producţiile EBNF, vă va ajuta dacă aveţi o copie tipărită şi mai multe creioane colorate astfel încât să puteţi urmări pe tot parcursul secvenţa exactă de înlocuiri pentru o anumită producţie. Urmaţi labirintul regulilor producţiilor până când ajungeţi la o frunză, un nod final, care nu conţine nume, ci numai text citat sau anumite reguli textuale.

Caracterele *, + şi ? au un înţeles special într-o producţie EBNF dacă apar după un element sau un grup de elemente încadrat de paranteze. Un asterisc (*) indică faptul că pot

exista zero sau mai multe instanțe ale acelui element. Un plus (+) spune că pot exista una sau mai multe instanțe ale acelui element. Iar un semn de întrebare (?) spune că pot exista zero instanțe sau o instanță a acelui element. Dacă nu există nici un sufix, înseamnă că un singur element este obligatoriu și nu poate fi repetat la acel nivel. Sunt aceleași convenții care sunt folosite și în cazul expresiilor regulate, un concept UNIX care își croiește drum către mintea și inima publicului, și în cazul documentelor XML DTD.

Marcajele XML sunt constrânse printr-o cerință care spune că, indiferent ce se va întâmpla cu o entitate de marcare prin substituții de entități și analize, rezultatul trebuie să fie complet în sine. Deci, dacă orice marcaj conține caracterul < cel mai din stânga al unui alt marcaj, trebuie să includă de asemenea și caracterul > cel mai din dreapta, care închide marcajul. Toate cuvintele cheie XML, cum sunt `ENTITY`, `NOTATION`, `ELEMENT` și altele, trebuie să fie scrise cu majuscule. Aici intră și tipurile de date și tipurile simbolice, cum sunt `#PCDATA`, `IDREF` și altele.

Declarația XML

Prin combinarea producțiilor 1, 22, 23 și 28, vedeți că orice DTD XML începe cu un prolog care poate să conțină sau nu un conținut relevant. Prologul poate să conțină o instrucțiune care, formal, este o comandă de procesare care se declară a fi XML și are trei atribute utile:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
```

Fișierele XML pot să omită această declarație, deși nu există un motiv întemeiat pentru a face așa ceva, iar W3C spune că ar trebui să o includeți. A fost declarată opțională numai din cauză că existau numeroase fișiere HTML și SGML care erau și perfect corecte XML. Nu avea nici un sens să excludă acele pagini.

Dacă doriți să specificați codificarea, cum de fapt ar trebui, este necesar să aveți un DTD. Ambele atribute, `encoding` și `standalone`, sunt, totuși, opționale. Procesoarele care întâlnesc `standalone="yes"` pot să ignore referințele externe în timpul validării documentului, ceea ce poate fi într-o oarecare măsură util dintr-un punct de vedere sau altul. Însă nu vă puteți baza pe un comportament sau altul, deoarece nu li se cere să facă neapărat ceva cu această informație.

Numărul de versiune este obligatoriu dacă folosiți declarația și el identifică versiunea XML pe care o respectă codul dumneavoastră. Procesoarele care văd o versiune pe care nu o suportă ar trebui să se oprească și să vă comunice acest lucru. În momentul actual există o singură versiune, 1.0, dar trebuie oricum să o specificați.

Codificarea este opțională, dar vă permite să specificați ce standarde pentru codificarea caracterelor sunt folosite în document. Cele mai obișnuite valori sunt „UTF-8", „UTF-16", „ISO-10646-UCS-2" și „ISO-10646-UCS-4", care sunt folosite pentru părți din Unicode / ISO/IEC 10646. „ISO-8859-1", „ISO-8859-2" și așa mai departe până la „ISO-8859-9" sunt folosite pentru spațiile de caractere ISO 8859, iar „ISO-2022-JP", „Shift_JIS" și „EUC-JP" pot fi folosite pentru spațiul de caractere japoneze JIS X-0208-1997. Lista completă a opțiunilor poate fi găsită în Recomandarea XML 1.0 de la W3C, `http://www.w3.org/XML/`. Anexa C, „Entități caracter", listează valorile entităților caracter ale ISO-8859-1 folosite în mod obișnuit în limbile vest-europene.

`standalone` vă permite să declarați dacă documentul folosește marcaje externe sau nu. Această informație este utilă pentru a permite procesorului să hotărască dacă va converti sau nu documentul într-o versiune autonomă, în cazul în care va fi trimis pentru a fi utilizat altundeva. Această instrucțiune conține o constrângere de validitate. Valoarea trebuie să fie `"no"` dacă este adevărată oricare din următoarele afirmații:

- Marcajul extern conține atribute cu valori prestabilite, iar documentul folosește elemente la care se aplică aceste atribute fără a seta explicit aceste valori în submulțimea internă a DTD-ului.
- Referiți în document entități caracter, altele decât `amp`, `lt`, `gt`, `apos` și `quot`, dar care nu sunt definite sau incluse prin referințe în submulțimea internă a DTD-ului.
- Marcajele dumneavoastră externe conțin orice atribute care trebuie *normalizate*, adică analizate, pentru a se rezolva referințele la entități și caractere și a se substitui un spațiu cu caractere spații albe.
- Există în documentul extern tipuri de elemente care conțin spații albe. Adică orice element de deschidere care urmează imediat după un alt element de deschidere, având între ele un retur de car sau un spațiu.

Instrucțiunea XML DOCTYPE

Prologul poate de asemenea să conțină o instrucțiune `DOCTYPE`. Fiecare DTD XML complet conține o singură instrucțiune `DOCTYPE` care denumește rădăcina arborelui de descriere a documentului XML (DTD-ul fiind definit) și include DTD-ul însuși ca submulțime internă sau externă, acestea fiind opționale:

```
<!DOCTYPE docname SYSTEM "numedocument.dtd" [
<!ELEMENT docname ( urmatorul(urmatoarele) niveluri ale arborelui ) >
… etc.
]>
```

Fragmentele DTD, cele care se presupune că se includ în alte DTD-uri, trebuie să omită această declarație, deoarece în fiecare DTD XML poate exista un singur `DOCTYPE`. În acest caz `DOCTYPE` poate fi declarat într-un alt fișier care include referințe la fragmentele DTD externe:

```
<!DOCTYPE docname SYSTEM "numedocument.dtd" [
<!ENTITY % subdtd1 SYSTEM "subdtdmeu1.dtd" >
%subdtd1;
<!ENTITY % subdtd2 SYSTEM "subdtdmeu2.dtd" >
%subdtd2;
]>
```

Observați că parantezele drepte care cuprind codul XML in-line delimitează submulțimea internă a întregului document XML care poate fi folosit pentru a suprascrie listele de atribute prestabilite și declarațiile de entități din submulțimea externă DTD. Acesta este mecanismul principal pentru modificarea și personalizarea DTD-urilor externe, deși puteți realiza acest lucru și din alt DTD.

Observați că rădăcina arborelui de descriere a documentului trebuie să aibă exact același nume cu `DOCTYPE`.

Entitatea XML

Vă puteți gândi la o `ENTITY` în XML ca la o facilitate macro dintr-un limbaj de programare, deși se supune unui set complex de reguli care stabilesc momentul și modul în care macroinstrucțiunea este executată sau interpretată. Tabelul A.1 vă prezintă o listă de referință rapidă a entităților, așa cum apare în Recomandarea XML 1.0 a W3C.

Tabelul A.1 Referință rapidă pentru entități

Tipul entității	Parametru	Generale interne	Generale externe analizate	Neanalizate	Caracter
Referință în conținut	Nu este recunoscut	Incluse	Incluse dacă se validează	Interzise	Incluse
Referință în valoarea unui atribut	Nu este recunoscut	Incluse în valori literale	Interzise	Interzise	Incluse
Apariții ca valoare a unui atribut	Nu este recunoscut	Interzise	Interzise	Notificate	Nu este recunoscut
Referință în valoarea unei entități	Incluse în valori literale	Derivate	Derivate	Interzise	Incluse
Referință în DTD	Incluse ca PE	Interzise	Interzise	Interzise	Interzise

Declarațiile ENTITY trebuie să existe deja atunci când sunt utilizate, deci ordinea în care sunt declarate entitățile în cascadă este critică. Plasamentul lor exact în DTD este de asemenea critic, deoarece entitățile sunt restricționate la submulțimea internă.

În câteva cuvinte, submulțimea internă a fost simplificată pentru ca analizoarele care nu validează, care oricum nu fac mare lucru cu DTD-ul, să nu trebuiască să fie prea inteligente. Pentru a atinge acest scop lăudabil, se permit destul de liber entitățile caracter dar se împiedică folosirea entităților parametru, cu excepția nivelului cel mai de sus, unde puteți utiliza marcaje.

Notațiile XML

Notațiile identifică formatul entităților care nu sunt analizate, elemente care conțin un atribut NOTATION și aplicațiile ajutătoare care primesc o instrucțiune de procesare. Utilizările tipice sunt instrucțiunile adresate unei aplicații referitoare la modul de manipulare a unor fișiere binare, cum ar fi grafică JPEG, sau ce trebuie făcut cu fragmentele de date, cum ar fi fișierele TeX sau Postscript.

Notațiile trebuie să fie declarate înainte de utilizare, o bună practică fiind să le includeți la începutul fișierului dumneavoastră. Viața vă va fi mai ușoară dacă grupați toate notațiile în același loc, dar acest lucru nu este obligatoriu. Mie îmi place să pun toate declarațiile ENTITY la început, apoi declarațiile NOTATION și apoi declarațiile ELEMENT. Prefer să grupez declarația ATTLIST cu declarația ELEMENT la care se aplică. Mi se pare logică această aranjare, deoarece așa pot să văd direct ce atribute se aplică la un element și știu exact unde să caut atributele când mă gândesc la un element, dar puteți folosi o cu totul altă grupare logică, dacă are sens pentru dumneavoastră.

Există două tipuri de declarații NOTATION. Primul vă permite să identificați modul în care anumite tipuri de obiecte sunt manipulate de o anumită aplicație care există pe sistemul dumneavoastră:

```
<!NOTATION nume-notatie SYSTEM "handler-date.exe ">
```


Poate fi utilă, de exemplu, pentru a declara că o aplicație trebuie să folosească Adobe Photoshop pentru a vizualiza fișiere GIF:

```
<!NOTATION gif SYSTEM "ps.exe ">
```

Puteți presupune că directorul PhotoShop se află în cale, dar la fel de bine ați putea folosi numele cu întreaga cale specificată. Desigur, faceți presupunerea că aplicația de asistență pe care o folosiți nu va fi schimbată și că oricine va folosi DTD-ul dumneavoastră va avea acces la aceeași aplicație de asistență. Aceste presupuneri nu sunt deosebit de sigure, dar pot fi justificate în anumite situații.

O variantă a acestei metode funcționează în Microsoft Windows și poate funcționează și în alte sisteme, dar nu se poate garanta:

```
<!NOTATION gif SYSTEM "GIF">
```

Prin aceasta, Windows caută în Registry o aplicație care poate manipula fișierele de tip „GIF" și apoi îi transferă informațiile.

O a doua strategie poate fi să vă referiți la un standard public, iar apoi să lăsați aplicația să își dea seama cum să folosească aceste informații pentru a manipula fișierul. Deci, în exemplul nostru GIF, veți folosi două declarații NOTATION care să manipuleze cele două tipuri de fișiere GIF:

```
<!NOTATION gif87a PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
87a//EN">
<!NOTATION gif89a PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
89a//EN">
```

Și această abordare are problemele sale. Deși aproape toate handlerele GIF recunosc automat cele două tipuri de GIF și se comportă adecvat versiunii, acum trebuie să știți tipul fișierelor GIF despre care vorbiți înainte de a le da referințele, în afara cazului în care aplicația noastră le manipulează pe amândouă. Ghinion. De asemenea, vă bazați pe bunele oficii ale registrului SGML, care, strict vorbind, nu există pentru XML decât în forma prestabilită, pentru a localiza aplicațiile din identificatorii publici. XML cere de fapt să însoțiți identificatorul public cu un URI care identifică o aplicație ce poate manipula acest tip de fișiere. Deci sintaxa va arăta astfel:

```
<!NOTATION gif87a PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
87a//EN" "ps.exe ">
<!NOTATION gif89a PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format
89a//EN" "ps.exe ">
```

După ce ați declarat o notație, iată cum o puteți folosi într-un atribut cu o listă enumerată:

```
<!ELEMENT  img EMPTY>
<!ATTLIST  img
           src %URL #REQUIRED
           alt CDATA #REQUIRED
           type NOTATION (gif89a | gif87a) "gif89a">
```

Iată, de asemenea, cum o puteți folosi ca tip de date prestabilit:

```
<!ELEMENT  img EMPTY>
<!ATTLIST  img
           src %URL #REQUIRED
           alt CDATA #REQUIRED
           type NDATA gif89a>
```

Orice nume de notație trebuie să fie declarat înainte de utilizare.

ELEMENT în XML

ELEMENT-ele sunt adesea analizate și aranjate într-o structură internă de arbore, de fapt un tabel de căutare, în momentul citirii lor. Asta înseamnă că ordinea în care ajung în fluxul de date nu este foarte importantă atâta timp cât nu există dubluri, acesta fiind o eroare fatală.

Fiecare arbore complet de descriere a documentului trebuie să aibă un singur element rădăcină. Numele acestui ELEMENT trebuie să fie exact același nume cu cel al DOCTYPE folosit la identificarea DTD-ului.

Observați că astfel nu este împiedicată includerea unui fragment DTD ca frunză într-un arbore, fie ca submulțime externă, fie ca entitate parametru.

Un ELEMENT cu conținut arată astfel:

```
<!ELEMENT numeelement (continut)>
```

unde în conținut poate fi orice număr de tipuri de date sau alte elemente date într-o sintaxă, care să vă permită să declarați un model de conținut care specifică elementele ce pot apărea, ordinea acestora și numărul permis de apariții.

Un ELEMENT fără conținut arată astfel:

```
<!ELEMENT numeelement EMPTY>
```

Un element vid se comportă aproape la fel cu un element HTML vid; adică folosește o singură etichetă cu atribute opționale pentru a face tot ce trebuie. Spre deosebire de HTML, totuși, în XML trebuie să închideți și etichetele vide, fie ca:

```
<elementvid />
```

fie ca:

```
<elementvid></elementvid>
```

Forma a doua este destul de confuză, deoarece, dacă vă uitați la el, puteți crede că este un element care poate primi un conținut. Vă recomand să folosiți eticheta cu o singură parte și cu slash-ul de închidere inclus.

Opusul unui element vid este un element fără restricții, care poate să aibă orice conținut. Sintaxa este simplă:

```
<!ELEMENT numeelement ANY>
```

După cum vă dați seama, acest tip de elemente nu poate facilita prea multe din punct de vedere al verificării validității.

Instrucțiunile de procesare XML

Comenzile de procesare XML vă permit să înglobați informații specifice unei anumite aplicații în interiorul documentului dumneavoastră XML. Orice conținut din instrucțiune este ignorat de XML și de orice procesor care nu este capabil să interpreteze comanda. Deci, de exemplu, iată cum ați putea să inserați un script în Perl în documentul dumneavoastră:

```
… <?perl use Date::DateCalc; if (Date::DateCalc::Version() eq "3.2") {print
"OK\n";} else {print "Fail\n ";} … ?> …
```

Dacă nu sunteți sigur ce procesoare există pe un anumit sistem țintă, puteți furniza mai multe opțiuni, deși nu este foarte clar ce se întâmplă dacă există mai multe procesoare.

Nu există (deocamdată) o sintaxă care să îi spună procesorului XML să ignore opțiunile dacă o instrucțiune de procesare a fost interpretată cu succes.

Comentariile XML

Comentariul XML este comentariul obișnuit din HTML și SGML, dar este limitat într-o oarecare măsură pentru a elimina posibilitatea includerii mai multor comentarii în același set de delimitatori. Deci, următorul comentariu este legal:

```
<!-- Text comentariu -->
```

iar acestea nu:

```
<!-- Text comentariu --    -- Al doilea comentariu -- -->
<!-- Text comentariu --
  -- Al doilea comentariu -->
```

deși ambele sunt corecte atât în SGML cât și în HTML 4.0.

Procesoarele XML nu sunt obligate să transfere comentariile nici unei aplicații externe, deci inserarea comenzilor de script sau de stil în comentarii nu este recomandată, deși în HTML este o practică des folosită. Pentru ascunderea datelor de procesorul XML, folosiți o secțiune CDATA sau o instrucțiune de procesare și nu un comentariu.

Secțiunile CDATA în XML

Dacă doriți să ascundeți un text de procesorul XML, secțiunea CDATA este ideală. Orice se află în interiorul secțiunii CDATA, este complet ignorat. Nu se disting entități, nu se recunosc marcaje și nu se face nimic în afara transferului textului exact așa cum este spre aplicație:

```
<![CDATA[Aici poate fi orice … ]]>
```

Secțiunile CDATA sunt utile atunci când doriți să transmiteți date care ar putea fi confundate cu XML, cum ar fi o etichetă HTML, sau să folosiți un text care conține caractere rezervate, cum ar fi semnele pentru ampersand (&) sau parantezele unghiulare (<) și, din anumite motive, nu doriți să le includeți în secvențe escape. Secvența]]> nu are voie să apară în textul inclus, deoarece secțiunea CDATA ar fi încheiată prea devreme.

Fișierul de catalogare XML

Deși atunci când folosiți identificatorii PUBLIC ar trebui să folosiți oficial sistemul cu citate al sintaxei identificatorilor, există un fișier moștenit SGML, numit fișier de catalogare, localizat în același director cu documentul XML sau într-o altă locație care depinde de aplicație. Dacă procesorul dumneavoastră XML suportă acest fișier, atunci va fi completat cu liste de identificatori PUBLIC și identificatori SYSTEM fără nume, pe care procesorul dumneavoastră îi poate folosi pentru a rezolva referințele la identificatori PUBLIC. Aceasta este o modalitate mai bună de a suporta referințele externe, deoarece dacă un fișier extern se mută sau se modifică, puteți actualiza toate referințele la el dintr-o mișcare și nu prin căutări obositoare ale referinței prin sute sau mii de documente și prin modificări repetate la fiecare apariție. Într-un mediu care obligă la controlul modificărilor, o „simplă" modificare ca cea de mai sus poate duce la săptămâni de muncă pentru verificarea tuturor documentelor.

Fișierul de catalogare arată cam așa:

```
-- catalog --
-- comments --
PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format 87a//EN" "ps.exe "
PUBLIC "-//CompuServe//NOTATION Graphics Interchange Format 89a//EN" "ps.exe "
PUBLIC "-//W3C//DTD XHTML Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/transitional.dtd"
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/transitional.dtd "
...
```

unde fiecare identificator `PUBLIC` este împerecheat cu un URI care definește o aplicație sau alte referințe externe care corespund. Este același tip de date pe care le folosiți într-o declarație `DOCTYPE` sau, de fapt, în sistemul cu citate al sintaxei identificatorilor XML menționat mai sus. Avantajul catalogului este că nu mai trebuie să tastați atâta și înlătură o posibilitate de apariție a erorilor:

```
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        http://www.w3.org/TR/xhtml1/DTD/transitional.dtd
```

Întreaga problemă a găsirii documentelor externe XML este încă neclară și, deși se lucrează pentru a menține locații stabile pe Web (vezi articolul despre URI-uri grozave la `http://www.w3.org/Provider/Style/URI`), problema denumirii și localizării documentelor este foarte dificilă. Aceasta cere cooperarea milioanelor de situri, majoritatea acestora fiind într-o stare constantă de haos și dezordine, noii administratori (sau administratorii vechi cu idei noi) schimbând cât ai bate din palme locațiile fișierelor.

Folosirea periodică a verificatorilor de legături este o idee bună, dar aceasta nu vă ajută să găsiți locația nouă. De fapt, multe situri nu afișează niciodată anunțuri de modificare la locația veche. Într-o demonstrație uluitoare de orgoliu și egocentrism, ei presupun că oricine va fi motivat să folosească motoare de căutare interne pentru a găsi unde se află acum pagina veche. Deci suntem cam optimiști dacă sperăm într-o cooperare binevoitoare.

Ca și restul comunității Web, XML va trebui să se descurce în continuare cu metode imperfecte de legare la documente, până când va avea loc un progres teoretic.

Între timp să studiem un DTD XHTML.

Referința tehnică XHTML

În carte am introdus doar DTD-ul tranzitoriu din motive de spațiu. Întregul set de DTD-uri este disponibil ca o legătură de la `http://www.w3.org/TR/xhtml1/`, dar DTD-urile efective pot să se modifice puțin, standardul evoluând.

DTD-ul tranzitoriu XHTML versiunea 1

DTD-ul tranzitoriu XHTML prezentat în Listingul A.2 este cel mai probabil să se potrivească și documentelor oarecare HTML. Deși nu asigură un suport pentru cadre, acestea oricum sunt destul de rare pe Web și sunt ferm descurajate, atât din cauza preferințelor utilizatorilor cât și din importanța care trebuie dată dificultăților pe care le pun în fața persoanelor cu handicap de vedere sau motor.

Listingul A.2 DTD-ul tranzitoriu XHTML 1.01

```
<!--
   Extensible HTML version 1.0 Transitional DTD
   This is the same as HTML 4.0 Transitional except for
   changes due to the differences between XML and SGML.
   Namespace = http://www.w3.org/1999/xhtml
   For further information, see: http://www.w3.org/TR/xhtml1
   Copyright (c) 1998-2000 W3C (MIT, INRIA, Keio),
   All Rights Reserved.
   This DTD module is identified by the PUBLIC and SYSTEM identifiers:
   PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
   $Revision: 1.1 $
   $Date: 2000/01/26 14:08:56 $

-->

<!--================= Character mnemonic entities =========================-->

<!ENTITY % HTMLlat1 PUBLIC
   "-//W3C//ENTITIES Latin 1 for XHTML//EN"
   "xhtml-lat1.ent">
%HTMLlat1;

<!ENTITY % HTMLsymbol PUBLIC
   "-//W3C//ENTITIES Symbols for XHTML//EN"
   "xhtml-symbol.ent">
%HTMLsymbol;

<!ENTITY % HTMLspecial PUBLIC
   "-//W3C//ENTITIES Special for XHTML//EN"
   "xhtml-special.ent">
%HTMLspecial;

<!--================== Imported Names ====================================-->

<!ENTITY % ContentType "CDATA">
    <!-- media type, as per [RFC2045] -->

<!ENTITY % ContentTypes "CDATA">
    <!-- comma-separated list of media types, as per [RFC2045] -->

<!ENTITY % Charset "CDATA">
    <!-- a character encoding, as per [RFC2045] -->

<!ENTITY % Charsets "CDATA">
    <!-- a space separated list of character encodings, as per [RFC2045] -->

<!ENTITY % LanguageCode "NMTOKEN">
    <!-- a language code, as per [RFC1766] -->

<!ENTITY % Character "CDATA">
    <!-- a single character from [ISO10646] -->
```

Listingul A.2 (continuare)

```
<!ENTITY % Number "CDATA">
    <!-- one or more digits -->

<!ENTITY % LinkTypes "CDATA">
    <!-- space-separated list of link types -->

<!ENTITY % MediaDesc "CDATA">
    <!-- single or comma-separated list of media descriptors -->

<!ENTITY % URI "CDATA">
    <!-- a Uniform Resource Identifier, see [RFC2396] -->

<!ENTITY % UriList "CDATA">
    <!-- a space separated list of Uniform Resource Identifiers -->

<!ENTITY % Datetime "CDATA">
    <!-- date and time information. ISO date format -->

<!ENTITY % Script "CDATA">
    <!-- script expression -->

<!ENTITY % StyleSheet "CDATA">
    <!-- style sheet data -->

<!ENTITY % Text "CDATA">
    <!-- used for titles etc. -->

<!ENTITY % FrameTarget "NMTOKEN">
    <!-- render in this frame -->

<!ENTITY % Length "CDATA">
    <!-- nn for pixels or nn% for percentage length -->

<!ENTITY % MultiLength "CDATA">
    <!-- pixel, percentage, or relative -->

<!ENTITY % MultiLengths "CDATA">
    <!-- comma-separated list of MultiLength -->

<!ENTITY % Pixels "CDATA">
    <!-- integer representing length in pixels -->

<!-- these are used for image maps -->

<!ENTITY % Shape "(rect|circle|poly|default)">

<!ENTITY % Coords "CDATA">
    <!-- comma separated list of lengths -->

<!-- used for object, applet, img, input and iframe -->
<!ENTITY % ImgAlign "(top|middle|bottom|left|right)">
```



```
<!-- a color using sRGB: #RRGGBB as Hex values -->
<!ENTITY % Color "CDATA">

<!-- There are also 16 widely known color names with their sRGB values:

    Black       = #000000     Green      = #008000
    Silver      = #C0C0C0     Lime       = #00FF00
    Gray        = #808080     Olive      = #808000
    White       = #FFFFFF     Yellow     = #FFFF00
    Maroon      = #800000     Navy       = #000080
    Red         = #FF0000     Blue       = #0000FF
    Purple      = #800080     Teal       = #008080
    Fuchsia     = #FF00FF     Aqua       = #00FFFF
-->

<!--==================== Generic Attributes ===============================-->

<!-- core attributes common to most elements
  id        document-wide unique id
  class     space separated list of classes
  style     associated style info
  title     advisory title/amplification
-->
<!ENTITY % coreattrs
 "id          ID              #IMPLIED
  class       CDATA           #IMPLIED
  style       %StyleSheet;    #IMPLIED
  title       %Text;          #IMPLIED"
  >

<!-- internationalization attributes
  lang        language code (backwards compatible)
  xml:lang    language code (as per XML 1.0 spec)
  dir         direction for weak/neutral text
-->
<!ENTITY % i18n
 "lang        %LanguageCode;  #IMPLIED
  xml:lang    %LanguageCode;  #IMPLIED
  dir         (ltr|rtl)       #IMPLIED"
  >

<!-- attributes for common UI events
   onclick       a pointer button was clicked
   ondblclick    a pointer button was double clicked
   onmousedown   a pointer button was pressed down
   onmouseup     a pointer button was released
   onmousemove   a pointer was moved onto the element
   onmouseout    a pointer was moved away from the element
   onkeypress    a key was pressed and released
   onkeydown     a key was pressed down
   onkeyup       a key was released
-->
```

Listingul A.2 (continuare)

```
<!ENTITY % events
 "onclick      %Script;       #IMPLIED
  ondblclick   %Script;       #IMPLIED
  onmousedown  %Script;       #IMPLIED
  onmouseup    %Script;       #IMPLIED
  onmouseover  %Script;       #IMPLIED
  onmousemove  %Script;       #IMPLIED
  onmouseout   %Script;       #IMPLIED
  onkeypress   %Script;       #IMPLIED
  onkeydown    %Script;       #IMPLIED
  onkeyup      %Script;       #IMPLIED"
  >

<!-- attributes for elements that can get the focus
  accesskey    accessibility key character
  tabindex     position in tabbing order
  onfocus      the element got the focus
  onblur       the element lost the focus
-->
<!ENTITY % focus
 "accesskey    %Character;    #IMPLIED
  tabindex     %Number;       #IMPLIED
  onfocus      %Script;       #IMPLIED
  onblur       %Script;       #IMPLIED"
  >

<!ENTITY % attrs "%coreattrs; %i18n; %events;">

<!-- text alignment for p, div, h1-h6. The default is
     align="left" for ltr headings, "right" for rtl -->

<!ENTITY % TextAlign "align (left|center|right) #IMPLIED">

<!--================== Text Elements ====================================-->

<!ENTITY % special
   "br | span | bdo | object | applet | img | map | iframe">

<!ENTITY % fontstyle "tt | i | b | big | small | u
                    | s | strike |font | basefont">

<!ENTITY % phrase "em | strong | dfn | code | q | sub | sup |
                   samp | kbd | var | cite | abbr | acronym">

<!ENTITY % inline.forms "input | select | textarea | label | button">

<!-- these can occur at block or inline level -->
<!ENTITY % misc "ins | del | script | noscript">

<!ENTITY % inline "a | %special; | %fontstyle; | %phrase; | %inline.forms;">

<!-- %Inline; covers inline or "text-level" elements -->
```



```
<!ENTITY % Inline "(#PCDATA | %inline; | %misc;)*">

<!--================== Block level elements ==============================-->

<!ENTITY % heading "h1|h2|h3|h4|h5|h6">
<!ENTITY % lists "ul | ol | dl | menu | dir">
<!ENTITY % blocktext "pre | hr | blockquote | address | center | noframes">

<!ENTITY % block
    "p | %heading; | div | %lists; | %blocktext; | isindex |fieldset | table">

<!ENTITY % Block "(%block; | form | %misc;)*">

<!-- %Flow; mixes Block and Inline and is used for list items etc. -->
<!ENTITY % Flow "(#PCDATA | %block; | form | %inline; | %misc;)*">

<!--================== Content models for exclusions =====================-->

<!-- a elements use %Inline; excluding a -->

<!ENTITY % a.content
   "(#PCDATA | %special; | %fontstyle; | %phrase; | %inline.forms; | %misc;)*">

<!-- pre uses %Inline excluding img, object, applet, big, small,
     sub, sup, font, or basefont -->

<!ENTITY % pre.content
   "(#PCDATA | a | br | span | bdo | map | tt | i | b | u | s |
     %phrase; | %inline.forms;)*">

<!-- form uses %Flow; excluding form -->

<!ENTITY % form.content "(#PCDATA | %block; | %inline; | %misc;)*">

<!-- button uses %Flow; but excludes a, form, form controls, iframe -->

<!ENTITY % button.content
   "(#PCDATA | p | %heading; | div | %lists; | %blocktext; |
     table | br | span | bdo | object | applet | img | map |
     %fontstyle; | %phrase; | %misc;)*">

<!--================ Document Structure ==================================-->

<!-- the namespace URI designates the document profile -->

<!ELEMENT html (head, body)>
<!ATTLIST html
  %i18n;
  xmlns        %URI;          #FIXED 'http://www.w3.org/1999/xhtml'
  >

<!--================ Document Head =======================================-->

<!ENTITY % head.misc "(script|style|meta|link|object|isindex)*">
```

Listingul A.2 (continuare)

```
<!-- content model is %head.misc; combined with a single
     title and an optional base element in any order -->

<!ELEMENT head (%head.misc;,
     ((title, %head.misc;, (base, %head.misc;)?) |
      (base, %head.misc;, (title, %head.misc;))))>

<!ATTLIST head
  %i18n;
  profile     %URI;          #IMPLIED
  >

<!-- The title element is not considered part of the flow of text.
       It should be displayed, for example as the page header or
       window title. Exactly one title is required per document.
    -->
<!ELEMENT title (#PCDATA)>
<!ATTLIST title %i18n;>

<!-- document base URI -->

<!ELEMENT base EMPTY>
<!ATTLIST base
  href        %URI;          #IMPLIED
  target      %FrameTarget;  #IMPLIED
  >

<!-- generic metainformation -->
<!ELEMENT meta EMPTY>
<!ATTLIST meta
  %i18n;
  http-equiv  CDATA          #IMPLIED
  name        CDATA          #IMPLIED
  content     CDATA          #REQUIRED
  scheme      CDATA          #IMPLIED
  >

<!--
  Relationship values can be used in principle:

   a) for document specific toolbars/menus when used
      with the link element in document head e.g.
        start, contents, previous, next, index, end, help
   b) to link to a separate style sheet (rel="stylesheet")
   c) to make a link to a script (rel="script")
   d) by stylesheets to control how collections of
      html nodes are rendered into printed documents
   e) to make a link to a printable version of this document
      e.g. a PostScript or PDF version (rel="alternate" media="print")
-->

<!ELEMENT link EMPTY>
<!ATTLIST link
  %attrs;
  charset     %Charset;      #IMPLIED
  href        %URI;          #IMPLIED
  hreflang    %LanguageCode; #IMPLIED
  type        %ContentType;  #IMPLIED
  rel         %LinkTypes;    #IMPLIED
  rev         %LinkTypes;    #IMPLIED
  media       %MediaDesc;    #IMPLIED
  target      %FrameTarget;  #IMPLIED
  >

<!-- style info, which may include CDATA sections -->
<!ELEMENT style (#PCDATA)>
<!ATTLIST style
  %i18n;
  type        %ContentType;  #REQUIRED
  media       %MediaDesc;    #IMPLIED
  title       %Text;         #IMPLIED
  xml:space   (preserve)     #FIXED 'preserve'
  >

<!-- script statements, which may include CDATA sections -->
<!ELEMENT script (#PCDATA)>
<!ATTLIST script
  charset     %Charset;      #IMPLIED
  type        %ContentType;  #REQUIRED
  language    CDATA          #IMPLIED
  src         %URI;          #IMPLIED
  defer       (defer)        #IMPLIED
  xml:space   (preserve)     #FIXED 'preserve'
  >

<!-- alternate content container for non script-based rendering -->

<!ELEMENT noscript %Flow;>
<!ATTLIST noscript
  %attrs;
  >

<!--========================= Frames ===========================-->

<!-- inline subwindow -->

<!ELEMENT iframe %Flow;>
<!ATTLIST iframe
  %coreattrs;
  longdesc    %URI;          #IMPLIED
  name        NMTOKEN        #IMPLIED
  src         %URI;          #IMPLIED
  frameborder (1|0)          "1"
  marginwidth %Pixels;       #IMPLIED
  marginheight %Pixels;      #IMPLIED
```

Listingul A.2 (continuare)

```
  scrolling     (yes|no|auto)      "auto"
  align         %ImgAlign;         #IMPLIED
  height        %Length;           #IMPLIED
  width         %Length;           #IMPLIED
  >

<!-- alternate content container for non frame-based rendering -->

<!ELEMENT noframes %Flow;>
<!ATTLIST noframes
  %attrs;
  >

<!--========================== Document Body =====================================-->

<!ELEMENT body %Flow;>
<!ATTLIST body
  %attrs;
  onload        %Script;           #IMPLIED
  onunload      %Script;           #IMPLIED
  background    %URI;              #IMPLIED
  bgcolor       %Color;            #IMPLIED
  text          %Color;            #IMPLIED
  link          %Color;            #IMPLIED
  vlink         %Color;            #IMPLIED
  alink         %Color;            #IMPLIED
  >

<!ELEMENT div %Flow;>  <!-- generic language/style container -->
<!ATTLIST div
  %attrs;
  %TextAlign;
  >

<!--========================== Paragraphs ========================================-->

<!ELEMENT p %Inline;>
<!ATTLIST p
  %attrs;
  %TextAlign;
  >

<!--========================== Headings ==========================================-->

<!--
  There are six levels of headings from h1 (the most important)
  to h6 (the least important).
-->

<!ELEMENT h1  %Inline;>
<!ATTLIST h1
  %attrs;
```



```
  %TextAlign;
  >

<!ELEMENT h2 %Inline;>
<!ATTLIST h2
  %attrs;
  %TextAlign;
  >

<!ELEMENT h3 %Inline;>
<!ATTLIST h3
  %attrs;
  %TextAlign;
  >

<!ELEMENT h4 %Inline;>
<!ATTLIST h4
  %attrs;
  %TextAlign;
  >

<!ELEMENT h5 %Inline;>
<!ATTLIST h5
  %attrs;
  %TextAlign;
  >

<!ELEMENT h6 %Inline;>
<!ATTLIST h6
  %attrs;
  %TextAlign;
  >

<!--========================== Lists =============================================-->

<!-- Unordered list bullet styles -->

<!ENTITY % ULStyle "(disc|square|circle)">

<!-- Unordered list -->

<!ELEMENT ul (li)+>
<!ATTLIST ul
  %attrs;
  type          %ULStyle;          #IMPLIED
  compact       (compact)          #IMPLIED
  >

<!-- Ordered list numbering style

     1   arabic numbers      1, 2, 3, ...
     a   lower alpha         a, b, c, ...
     A   upper alpha         A, B, C, ...
```

Listingul A.2 (continuare)

```
    i   lower roman          i, ii, iii, ...
    I   upper roman          I, II, III, ...

    The style is applied to the sequence number which by default
    is reset to 1 for the first list item in an ordered list.
-->
<!ENTITY % OLStyle "CDATA">

<!-- Ordered (numbered) list -->

<!ELEMENT ol (li)+>
<!ATTLIST ol
  %attrs;
  type          %OLStyle;          #IMPLIED
  compact       (compact)          #IMPLIED
  start         %Number;           #IMPLIED
  >

<!-- single column list (DEPRECATED) -->
<!ELEMENT menu (li)+>
<!ATTLIST menu
  %attrs;
  compact       (compact)          #IMPLIED
  >

<!-- multiple column list (DEPRECATED) -->
<!ELEMENT dir (li)+>
<!ATTLIST dir
  %attrs;
  compact       (compact)          #IMPLIED
  >

<!-- LIStyle is constrained to: "(%ULStyle;|%OLStyle;)" -->
<!ENTITY % LIStyle "CDATA">

<!-- list item -->

<!ELEMENT li %Flow;>
<!ATTLIST li
  %attrs;
  type          %LIStyle;          #IMPLIED
  value         %Number;           #IMPLIED
  >

<!-- definition lists - dt for term, dd for its definition -->

<!ELEMENT dl (dt|dd)+>
<!ATTLIST dl
  %attrs;
  compact       (compact)          #IMPLIED
  >
```



```
<!ELEMENT dt %Inline;>
<!ATTLIST dt
  %attrs;
  >

<!ELEMENT dd %Flow;>
<!ATTLIST dd
  %attrs;
  >

<!--========================== Address =============================================-->

<!-- information on author -->

<!ELEMENT address %Inline;>
<!ATTLIST address
  %attrs;
  >

<!--========================== Horizontal Rule =====================================-->

<!ELEMENT hr EMPTY>
<!ATTLIST hr
  %attrs;
  align         (left|center|right)     #IMPLIED
  noshade       (noshade)               #IMPLIED
  size          %Pixels;                #IMPLIED
  width         %Length;                #IMPLIED
  >

<!--========================== Preformatted Text ===================================-->

<!-- content is %Inline; excluding
       "img|object|applet|big|small|sub|sup|font|basefont" -->

<!ELEMENT pre %pre.content;>
<!ATTLIST pre
  %attrs;
  width         %Number;           #IMPLIED
  xml:space     (preserve)         #FIXED 'preserve'
  >

<!--========================== Block-like Quotes ===================================-->

<!ELEMENT blockquote %Flow;>
<!ATTLIST blockquote
  %attrs;
  cite          %URI;              #IMPLIED
  >

<!--========================== Text alignment ======================================-->

<!-- center content -->
```

Listingul A.2 (continuare)

```
<!ELEMENT center %Flow;>
<!ATTLIST center
  %attrs;
  >

<!--========================== Inserted/Deleted Text ============================-->

<!--
  ins/del are allowed in block and inline content, but its
  inappropriate to include block content within an ins element
  occurring in inline content.
-->
<!ELEMENT ins %Flow;>
<!ATTLIST ins
  %attrs;
  cite          %URI;              #IMPLIED
  datetime      %Datetime;         #IMPLIED
  >

<!ELEMENT del %Flow;>
<!ATTLIST del
  %attrs;
  cite          %URI;              #IMPLIED
  datetime      %Datetime;         #IMPLIED
  >

<!--========================== The Anchor Element ================================-->

<!-- content is %Inline; except that anchors shouldn't be nested -->

<!ELEMENT a %a.content;>
<!ATTLIST a
  %attrs;
  charset       %Charset;          #IMPLIED
  type          %ContentType;      #IMPLIED
  name          NMTOKEN            #IMPLIED
  href          %URI;              #IMPLIED
  hreflang      %LanguageCode;     #IMPLIED
  rel           %LinkTypes;        #IMPLIED
  rev           %LinkTypes;        #IMPLIED
  accesskey     %Character;        #IMPLIED
  shape         %Shape;            "rect"
  coords        %Coords;           #IMPLIED
  tabindex      %Number;           #IMPLIED
  onfocus       %Script;           #IMPLIED
  onblur        %Script;           #IMPLIED
  target        %FrameTarget;      #IMPLIED
  >

<!--========================== Inline Elements ===================================-->

<!ELEMENT span %Inline;> <!-- generic language/style container -->
```



```
<!ATTLIST span
  %attrs;
  >

<!ELEMENT bdo %Inline;>  <!-- I18N BiDi over-ride -->
<!ATTLIST bdo
  %coreattrs;
  %events;
  lang          %LanguageCode;     #IMPLIED
  xml:lang      %LanguageCode;     #IMPLIED
  dir           (ltr|rtl)          #REQUIRED
  >

<!ELEMENT br EMPTY>   <!-- forced line break -->
<!ATTLIST br
  %coreattrs;
  clear         (left|all|right|none) "none"
  >

<!ELEMENT em %Inline;>   <!-- emphasis -->
<!ATTLIST em %attrs;>

<!ELEMENT strong %Inline;>   <!-- strong emphasis -->
<!ATTLIST strong %attrs;>

<!ELEMENT dfn %Inline;>   <!-- definitional -->
<!ATTLIST dfn %attrs;>

<!ELEMENT code %Inline;>   <!-- program code -->
<!ATTLIST code %attrs;>

<!ELEMENT samp %Inline;>   <!-- sample -->
<!ATTLIST samp %attrs;>

<!ELEMENT kbd %Inline;>  <!-- something user would type -->
<!ATTLIST kbd %attrs;>

<!ELEMENT var %Inline;>   <!-- variable -->
<!ATTLIST var %attrs;>

<!ELEMENT cite %Inline;>   <!-- citation -->
<!ATTLIST cite %attrs;>

<!ELEMENT abbr %Inline;>   <!-- abbreviation -->
<!ATTLIST abbr %attrs;>

<!ELEMENT acronym %Inline;>   <!-- acronym -->
<!ATTLIST acronym %attrs;>

<!ELEMENT q %Inline;>   <!-- inlined quote -->
<!ATTLIST q
  %attrs;
  cite          %URI;              #IMPLIED
  >
```

Listingul A.2 (continuare)

```
<!ELEMENT sub %Inline;> <!-- subscript -->
<!ATTLIST sub %attrs;>

<!ELEMENT sup %Inline;> <!-- superscript -->
<!ATTLIST sup %attrs;>

<!ELEMENT tt %Inline;>   <!-- fixed pitch font -->
<!ATTLIST tt %attrs;>

<!ELEMENT i %Inline;>   <!-- italic font -->
<!ATTLIST i %attrs;>

<!ELEMENT b %Inline;>   <!-- bold font -->
<!ATTLIST b %attrs;>

<!ELEMENT big %Inline;>   <!-- bigger font -->
<!ATTLIST big %attrs;>

<!ELEMENT small %Inline;>   <!-- smaller font -->
<!ATTLIST small %attrs;>

<!ELEMENT u %Inline;>   <!-- underline -->
<!ATTLIST u %attrs;>

<!ELEMENT s %Inline;>   <!-- strike-through -->
<!ATTLIST s %attrs;>

<!ELEMENT strike %Inline;>   <!-- strike-through -->
<!ATTLIST strike %attrs;>

<!ELEMENT basefont EMPTY>  <!-- base font size -->
<!ATTLIST basefont
  id          ID              #IMPLIED
  size        CDATA           #REQUIRED
  color       %Color;         #IMPLIED
  face        CDATA           #IMPLIED
  >

<!ELEMENT font %Inline;> <!-- local change to font -->
<!ATTLIST font
  %coreattrs;
  %i18n;
  size        CDATA           #IMPLIED
  color       %Color;         #IMPLIED
  face        CDATA           #IMPLIED
  >

<!--========================== Object ======================================-->
<!--
  object is used to embed objects as part of HTML pages.
  param elements should precede other content. Parameters
  can also be expressed as attribute/value pairs on the
```



```
  object element itself when brevity is desired.
-->

<!ELEMENT object (#PCDATA | param | %block; | form | %inline; | %misc;)*>
<!ATTLIST object
  %attrs;
  declare     (declare)         #IMPLIED
  classid     %URI;             #IMPLIED
  codebase    %URI;             #IMPLIED
  data        %URI;             #IMPLIED
  type        %ContentType;     #IMPLIED
  codetype    %ContentType;     #IMPLIED
  archive     %UriList;         #IMPLIED
  standby     %Text;            #IMPLIED
  height      %Length;          #IMPLIED
  width       %Length;          #IMPLIED
  usemap      %URI;             #IMPLIED
  name        NMTOKEN           #IMPLIED
  tabindex    %Number;          #IMPLIED
  align       %ImgAlign;        #IMPLIED
  border      %Pixels;          #IMPLIED
  hspace      %Pixels;          #IMPLIED
  vspace      %Pixels;          #IMPLIED
  >

<!--
  param is used to supply a named property value.
  In XML it would seem natural to follow RDF and support an
  abbreviated syntax where the param elements are replaced
  by attribute value pairs on the object start tag.
-->
<!ELEMENT param EMPTY>
<!ATTLIST param
  id          ID              #IMPLIED
  name        CDATA           #REQUIRED
  value       CDATA           #IMPLIED
  valuetype   (data|ref|object) "data"
  type        %ContentType;   #IMPLIED
  >

<!--========================== Java applet ==================================-->
<!--
  One of code or object attributes must be present.
  Place param elements before other content.
-->
<!ELEMENT applet (#PCDATA | param | %block; | form | %inline; | %misc;)*>
<!ATTLIST applet
  %coreattrs;
  codebase    %URI;             #IMPLIED
  archive     CDATA             #IMPLIED
  code        CDATA             #IMPLIED
  object      CDATA             #IMPLIED
  alt         %Text;            #IMPLIED
```

Listingul A.2 (continuare)

```
  name          NMTOKEN          #IMPLIED
  width         %Length;         #REQUIRED
  height        %Length;         #REQUIRED
  align         %ImgAlign;       #IMPLIED
  hspace        %Pixels;         #IMPLIED
  vspace        %Pixels;         #IMPLIED
  >

<!--========================== Images ============================================-->

<!--
   To avoid accessibility problems for people who aren't
   able to see the image, you should provide a text
   description using the alt and longdesc attributes.
   In addition, avoid the use of server-side image maps.
-->

<!ELEMENT img EMPTY>
<!ATTLIST img
  %attrs;
  src           %URI;            #REQUIRED
  alt           %Text;           #REQUIRED
  name          NMTOKEN          #IMPLIED
  longdesc      %URI;            #IMPLIED
  height        %Length;         #IMPLIED
  width         %Length;         #IMPLIED
  usemap        %URI;            #IMPLIED
  ismap         (ismap)          #IMPLIED
  align         %ImgAlign;       #IMPLIED
  border        %Length;         #IMPLIED
  hspace        %Pixels;         #IMPLIED
  vspace        %Pixels;         #IMPLIED
  >

<!-- usemap points to a map element which may be in this document
  or an external document, although the latter is not widely supported -->

<!--========================== Client-side image maps ============================-->

<!-- These can be placed in the same document or grouped in a
     separate document although this isn't yet widely supported -->

<!ELEMENT map ((%block; | form | %misc;)+ | area+)>
<!ATTLIST map
  %i18n;
  %events;
  id            ID               #REQUIRED
  class         CDATA            #IMPLIED
  style         %StyleSheet;     #IMPLIED
  title         %Text;           #IMPLIED
  name          CDATA            #IMPLIED
  >

<!ELEMENT area EMPTY>
<!ATTLIST area
  %attrs;
  shape         %Shape;          „rect“
  coords        %Coords;         #IMPLIED
  href          %URI;            #IMPLIED
  nohref        (nohref)         #IMPLIED
  alt           %Text;           #REQUIRED
  tabindex      %Number;         #IMPLIED
  accesskey     %Character;      #IMPLIED
  onfocus       %Script;         #IMPLIED
  onblur        %Script;         #IMPLIED
  target        %FrameTarget;    #IMPLIED
  >

<!--========================== Forms =============================================-->

<!ELEMENT form %form.content;>   <!-- forms shouldn't be nested -->

<!ATTLIST form
  %attrs;
  action        %URI;            #REQUIRED
  method        (get|post)       "get"
  name          NMTOKEN          #IMPLIED
  enctype       %ContentType;    "application/x-www-form-urlencoded"
  onsubmit      %Script;         #IMPLIED
  onreset       %Script;         #IMPLIED
  accept        %ContentTypes;   #IMPLIED
  accept-charset %Charsets;      #IMPLIED
  target        %FrameTarget;    #IMPLIED
  >

<!--
  Each label must not contain more than ONE field
  Label elements shouldn't be nested.
-->
<!ELEMENT label %Inline;>
<!ATTLIST label
  %attrs;
  for           IDREF            #IMPLIED
  accesskey     %Character;      #IMPLIED
  onfocus       %Script;         #IMPLIED
  onblur        %Script;         #IMPLIED
  >

<!ENTITY % InputType
  "(text | password | checkbox |
    radio | submit | reset |
    file | hidden | image | button)"
   >

<!-- the name attribute is required for all but submit & reset -->
```

Listingul A.2 (continuare)

```
<!ELEMENT input EMPTY>      <!-- form control -->
<!ATTLIST input
  %attrs;
  type          %InputType;       "text"
  name          CDATA             #IMPLIED
  value         CDATA             #IMPLIED
  checked       (checked)         #IMPLIED
  disabled      (disabled)        #IMPLIED
  readonly      (readonly)        #IMPLIED
  size          CDATA             #IMPLIED
  maxlength     %Number;          #IMPLIED
  src           %URI;             #IMPLIED
  alt           CDATA             #IMPLIED
  usemap        %URI;             #IMPLIED
  tabindex      %Number;          #IMPLIED
  accesskey     %Character;       #IMPLIED
  onfocus       %Script;          #IMPLIED
  onblur        %Script;          #IMPLIED
  onselect      %Script;          #IMPLIED
  onchange      %Script;          #IMPLIED
  accept        %ContentTypes;    #IMPLIED
  align         %ImgAlign;        #IMPLIED
  >

<!ELEMENT select (optgroup|option)+>  <!-- option selector -->
<!ATTLIST select
  %attrs;
  name          CDATA             #IMPLIED
  size          %Number;          #IMPLIED
  multiple      (multiple)        #IMPLIED
  disabled      (disabled)        #IMPLIED
  tabindex      %Number;          #IMPLIED
  onfocus       %Script;          #IMPLIED
  onblur        %Script;          #IMPLIED
  onchange      %Script;          #IMPLIED
  >

<!ELEMENT optgroup (option)+>   <!-- option group -->
<!ATTLIST optgroup
  %attrs;
  disabled      (disabled)        #IMPLIED
  label         %Text;            #REQUIRED
  >

<!ELEMENT option (#PCDATA)>     <!-- selectable choice -->
<!ATTLIST option
  %attrs;
  selected      (selected)        #IMPLIED
  disabled      (disabled)        #IMPLIED
  label         %Text;            #IMPLIED
  value         CDATA             #IMPLIED
  >
```



```
<!ELEMENT textarea (#PCDATA)>     <!-- multi-line text field -->
<!ATTLIST textarea
  %attrs;
  name          CDATA             #IMPLIED
  rows          %Number;          #REQUIRED
  cols          %Number;          #REQUIRED
  disabled      (disabled)        #IMPLIED
  readonly      (readonly)        #IMPLIED
  tabindex      %Number;          #IMPLIED
  accesskey     %Character;       #IMPLIED
  onfocus       %Script;          #IMPLIED
  onblur        %Script;          #IMPLIED
  onselect      %Script;          #IMPLIED
  onchange      %Script;          #IMPLIED
  >

<!--
  The fieldset element is used to group form fields.
  Only one legend element should occur in the content
  and if present should only be preceded by whitespace.
-->
<!ELEMENT fieldset (#PCDATA | legend | %block; | form | %inline; | %misc;)*>
<!ATTLIST fieldset
  %attrs;
  >

<!ENTITY % LAlign "(top|bottom|left|right)">

<!ELEMENT legend %Inline;>      <!-- fieldset label -->
<!ATTLIST legend
  %attrs;
  accesskey     %Character;       #IMPLIED
  align         %LAlign;          #IMPLIED
  >

<!--
 Content is %Flow; excluding a, form, form controls, iframe
-->
<!ELEMENT button %button.content;>  <!-- push button -->
<!ATTLIST button
  %attrs;
  name          CDATA                   #IMPLIED
  value         CDATA                   #IMPLIED
  type          (button|submit|reset)   "submit"
  disabled      (disabled)              #IMPLIED
  tabindex      %Number;                #IMPLIED
  accesskey     %Character;             #IMPLIED
  onfocus       %Script;                #IMPLIED
  onblur        %Script;                #IMPLIED
  >

<!-- single-line text input control (DEPRECATED) -->
<!ELEMENT isindex EMPTY>
```

Listingul A.2 (continuare)

```
<!ATTLIST isindex
  %coreattrs;
  %i18n;
  prompt       %Text;            #IMPLIED
  >

<!--========================== Tables =================================-->

<!-- Derived from IETF HTML table standard, see [RFC1942] -->

<!--
 The border attribute sets the thickness of the frame around the
 table. The default units are screen pixels.

 The frame attribute specifies which parts of the frame around
 the table should be rendered. The values are not the same as
 CALS to avoid a name clash with the valign attribute.
-->
<!ENTITY % TFrame "(void|above|below|hsides|lhs|rhs|vsides|box|border)">

<!--
 The rules attribute defines which rules to draw between cells:

 If rules is absent then assume:
     "none" if border is absent or border="0" otherwise "all"
-->

<!ENTITY % TRules "(none | groups | rows | cols | all)">

<!-- horizontal placement of table relative to document -->
<!ENTITY % TAlign "(left|center|right)">

<!-- horizontal alignment attributes for cell contents

  char         alignment char, e.g. char=':'
  charoff      offset for alignment char
-->
<!ENTITY % cellhalign
  "align       (left|center|right|justify|char) #IMPLIED
   char        %Character;       #IMPLIED
   charoff     %Length;          #IMPLIED"
  >

<!-- vertical alignment attributes for cell contents -->
<!ENTITY % cellvalign
  "valign      (top|middle|bottom|baseline) #IMPLIED"
  >

<!ELEMENT table
     (caption?, (col*|colgroup*), thead?, tfoot?, (tbody+|tr+))>
<!ELEMENT caption  %Inline;>
<!ELEMENT thead    (tr)+>
<!ELEMENT tfoot    (tr)+>
<!ELEMENT tbody    (tr)+>
<!ELEMENT colgroup (col)*>
<!ELEMENT col      EMPTY>
<!ELEMENT tr       (th|td)+>
<!ELEMENT th       %Flow;>
<!ELEMENT td       %Flow;>

<!ATTLIST table
  %attrs;
  summary      %Text;            #IMPLIED
  width        %Length;          #IMPLIED
  border       %Pixels;          #IMPLIED
  frame        %TFrame;          #IMPLIED
  rules        %TRules;          #IMPLIED
  cellspacing  %Length;          #IMPLIED
  cellpadding  %Length;          #IMPLIED
  align        %TAlign;          #IMPLIED
  bgcolor      %Color;           #IMPLIED
  >

<!ENTITY % CAlign "(top|bottom|left|right)">

<!ATTLIST caption
  %attrs;
  align        %CAlign;          #IMPLIED
  >

<!--
colgroup groups a set of col elements. It allows you to group
several semantically related columns together.
-->
<!ATTLIST colgroup
  %attrs;
  span         %Number;          "1"
  width        %MultiLength;     #IMPLIED
  %cellhalign;
  %cellvalign;
  >

<!--
 col elements define the alignment properties for cells in
 one or more columns.

 The width attribute specifies the width of the columns, e.g.

     width=64                   width in screen pixels
     width=0.5*                 relative width of 0.5

 The span attribute causes the attributes of one
 col element to apply to more than one column.
-->
<!ATTLIST col
```

Listingul A.2 (continuare)

```
  %attrs;
  span          %Number;          "1"
  width         %MultiLength;     #IMPLIED
  %cellhalign;
  %cellvalign;
  >

<!--
    Use thead to duplicate headers when breaking table
    across page boundaries, or for static headers when
    tbody sections are rendered in scrolling panel.

    Use tfoot to duplicate footers when breaking table
    across page boundaries, or for static footers when
    tbody sections are rendered in scrolling panel.

    Use multiple tbody sections when rules are needed
    between groups of table rows.
-->
<!ATTLIST thead
  %attrs;
  %cellhalign;
  %cellvalign;
  >

<!ATTLIST tfoot
  %attrs;
  %cellhalign;
  %cellvalign;
  >

<!ATTLIST tbody
  %attrs;
  %cellhalign;
  %cellvalign;
  >

<!ATTLIST tr
  %attrs;
  %cellhalign;
  %cellvalign;
  bgcolor       %Color;           #IMPLIED
  >

<!-- Scope is simpler than headers attribute for common tables -->
<!ENTITY % Scope "(row|col|rowgroup|colgroup)">

<!-- th is for headers, td for data and for cells acting as both -->

<!ATTLIST th
  %attrs;
  abbr          %Text;            #IMPLIED
```



```
  axis          CDATA             #IMPLIED
  headers       IDREFS            #IMPLIED
  scope         %Scope;           #IMPLIED
  rowspan       %Number;          "1"
  colspan       %Number;          "1"
  %cellhalign;
  %cellvalign;
  nowrap        (nowrap)          #IMPLIED
  bgcolor       %Color;           #IMPLIED
  width         %Pixels;          #IMPLIED
  height        %Pixels;          #IMPLIED
  >

<!ATTLIST td
  %attrs;
  abbr          %Text;            #IMPLIED
  axis          CDATA             #IMPLIED
  headers       IDREFS            #IMPLIED
  scope         %Scope;           #IMPLIED
  rowspan       %Number;          "1"
  colspan       %Number;          "1"
  %cellhalign;
  %cellvalign;
  nowrap        (nowrap)          #IMPLIED
  bgcolor       %Color;           #IMPLIED
  width         %Pixels;          #IMPLIED
  height        %Pixels;          #IMPLIED
  >
```

Note la fițuica XHTML

XHTML 1.0 a fost creat pentru a arăta și funcționa cât mai aproape de HTML 4.0. Deoarece HTML este o aplicație a SGML, în timp ce XHTML este o aplicație a XML, unele lucruri care erau permise în HTML 4.0 nu sunt permise în XHTML 1.0. În mod ciudat, și opusul este adevărat.

SGML permite impunerea unor obligații numite în SGML *excluderi*, care vă permit să declarați pur și simplu că anumite elemente nu sunt permise în interiorul altor elemente. XML nu permite acest nivel de simplitate, deoarece de fapt este foarte complicat de efectuat. O etichetă ancoră (`<A ...> ... </A>`), de exemplu, nu este permis să fie imbricată în altă ancoră în HTML 4.0. XHTML 1.0 se bazează pe bunăvoința și intențiile bune ale proiectantului XHTML pentru a nu face acest lucru în XHTML 1.0. Este într-o oarecare măsură nesatisfăcător, dar trebuie să ne mulțumească pentru moment. Aceste puncte în care pot scăpa erori pot fi întărite prin verificatori de sintaxă XHTML externi.

O mare schimbare de la HTML este că XHTML, ca și XML, este sensibil la tipul literelor. DTD-ul XHTML folosește de la un capăt la altul litere mici pentru numele de etichete și atribute și nu va accepta etichete sau atribute cu majuscule.

Toate atributele trebuie să fie încadrate în semne de citat și vor fi tratate ca și caractere. Orice procesare a valorilor cade în sarcina aplicației.

Orice etichetă trebuie să fie închisă corect, chiar și etichetele vide. Pentru compatibilitate, sintaxa sugerată pentru etichetele vide este adăugarea unui spațiu și a unui slash (/)

la sfârșitul etichetei vide. Deci echivalentul corect XHTML pentru <br> din HTML este
 și așa mai departe. Etichetele de paragraf trebuie închise, ceea ce puțini dintre noi o fac. Vă recomand să folosiți un bun verificator de sintaxă XML, cum ar fi HTML-Kit / Tidy pentru a insera automat etichetele de încheiere corecte, deoarece poate fi destul de obositor să le inserați manual. De asemenea, Tidy vă va converti toate etichetele și atributele la litere mici, eliminând o sarcină neplăcută pentru aceia dintre noi care au o grămadă de cod, peste tot în lume.

Sugestie de la LeeAnne.com Words to weave by

Programul Tidy al lui Dave Raggett este un program în linie de comandă care validează și opțional convertește codul HTML moștenit într-un echivalent XHTML. Poate fi găsit la `http://www.w3.org/ People/Raggett/tidy/` dar și încorporat în editorul cu interfață grafică (pentru Windows) HTML-Kit, la `http://www.chami.com/`.

Denumirea zonei de nume XHTML

Printre primele declarații `ELEMENT` pe care le vedeți în DTD-ul XHTML de mai sus este definiția etichetei `<html>` care ar trebui să includă orice document HTML sau XHTML. Observați că atributul zonei de nume `xmlns` este pre-declarat `FIXED`. Asta înseamnă că nu puteți suprascrie această valoare, chiar dacă încercați să furnizați o altă valoare pentru zona de nume. Are aproximativ același efect cu introducerea ei manuală astfel:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Atributul `xmlns` ar trebui întotdeauna să fie adăugat etichetei rădăcină. De fapt este obligatoriu, iar această versiune a DTD-ului XHTML are grijă de asta în locul dumneavoastră. În cazul altor DTD-uri, poate va fi nevoie să îl introduceți dumneavoastră, iar în cazul unor DTD-uri care se află în mai multe locații nu vă este permis luxul să îl declarați `FIXED`.

Cu atributul `xmlns` identificați zona de nume a documentului. În viitor, când în fiecare document vor fi suportate mai multe zone de nume, zona de nume va ajuta aplicațiile să acceseze modelul obiectual al paginii iar prezența unor zone de nume valide va facilita acest acces și alte prelucrări.

Recursivitate

Dacă începeți să desfaceți DTD-ul XHTML, veți observa unele declarații `ENTITY` care stabilesc ceea ce se cheamă modele de conținut pentru excludere. Țineți minte că am spus că eticheta ancoră era una dintre cele cu probleme? Ei bine, există o `ENTITY` numită `%a.content`, care stabilește o listă de elemente care pot fi incluse într-o etichetă ancoră. Pare că `<a>` nu este printre ele, fiind exclusă foarte corect prin lista de conținut specială. Dar stați puțin! Să vedem ce etichete apar în listă. Să alegem una la întâmplare, să zicem `<i>`. Dacă ne uităm la conținutul permis pentru această etichetă, `%Inline`, vedem că putem include în interiorul său o etichetă ancoră. Majoritatea includerilor potențiale ale etichetei ancoră sunt similare. Deci construcția incorectă de mai jos este evident ilegală și va fi interceptată de analizor:

```
… <a href=" … "><a href=" … "></a></a> …
```

Acest fragment va fi interceptat atunci când documentul este validat, dar vărul său bun, care de asemenea este ilegal, probabil nu va fi interceptat:


```
… <a href=" … "><i><a href=" … "></a></i></a> …
```

Pare să fie corect din punctul de vedere al DTD-ului XHTML.

Aceasta este o problemă generală a XML și este una care cred că va trebui să fie rezolvată într-o bună zi, dar a fost discutată pe larg de mult timp și decizia a fost să se implementeze în acest fel.

Extensibilitate

Acest DTD este un model din care puteți învăța ce trebuie să faceți pentru ca DTD-urile dumneavoastră să poată fi extinse ușor, fie de alții, fie de dumneavoastră dacă doriți să experimentați o caracteristică fără să modificați efectiv codul.

Fiecare structură importantă este descrisă într-o `ENTITY` înainte de a fi declarată, astfel încât să poată fi suprascrisă cu ușurință printr-un alt DTD, fie în submulțimea internă, fie ca DTD complet separat care apelează la DTD-ul XHTML după ce face propriile sale declarații.

Este bine structurat, toate declarațiile `ENTITY` fiind la începutul fișierului, bine comentat, iar declarațiile `ELEMENT` sunt grupate după funcție și sunt urmate imediat de declarațiile `ATTLIST` asociate lor.

Instrumente pentru editarea şi transformarea XML/XHTML

ANEXA

B

În această anexă

Validatoare XML
Editoare XML
Browsere cu suport XML
Instrumente de dezvoltare XML
Motoare de analizare XML
Instrumente de conversie XML/SGML
Instrumente XML diverse
Aplicaţii cu suport XML

Această anexă vă prezintă o listă de instrumente pe care poate le veţi considera utile atunci când lucraţi în XML şi în standardele legate de acesta. Gama variază de la programe simple în linie de comandă donate comunităţii dezvoltatorilor XML de creatori preocupaţi de binele public – şi de obicei foarte talentaţi – până la medii sofisticate de editare grafică pentru documente şi DTD-uri, create de corporaţii importante. În multe cazuri, descrierea unui instrument este însoţită de un exemplu al programului folosit sau de o descriere scurtă a modului de funcţionare.

Această anexă este împărţită în secţiuni clasificate după natura generală a instrumentelor, deşi uneori aceste categorii se suprapun. Dacă se întâmplă aşa, utilitatea va fi listată la ambele categorii.

Validatoare XML

Un validator XML verifică construirea corectă a documentului XML faţă de DTD-ul care îl defineşte, pentru a afla dacă documentul respectă toate regulile. În caz contrar, returnează o eroare fatală. În unele cazuri, validarea poate fi dezactivată pentru a uşura verificarea iniţială a unui document nou. Iată câteva validatoare bune:

- TclXML – Un analizor care validează, scris în limbajul de programare Tcl.

`http://www.zveno.com/zm.cgi/in-tclxml/`

- Xerces-J – Xerces-J este analizorul XML open-source, scris în Java, al companiei Apache Project.

`http://xml.apache.org/xerces-j/`

- fxp – Analizorul funcţional XML de la Universitatea din Trier, Germania. Este un analizor XML scris în Standard ML. Suportă XML Catalog şi conţine un modul de vizualizare a arborelui documentului.

`http://www.informatik.uni-trier.de/~neumann/Fxp/`

- Ælfred – Un analizor XML care validează, simplu, bazat pe Java, şi care este gratuit pentru utilizări ne-comerciale.

`http://www.opentext.com/services/content_management_services/xml_sgml_solutions.html#aelfred_and_sax`

Editoare XML

Editoarele XML vă permit să realizaţi conţinut XML într-un mediu prietenos cu utilizatorul. Deşi este posibil să creaţi XML într-un editor de text, suportul suplimentar, asigurat de un bun mediu de creare, face ca erorile să apară mult mai greu. Lista care urmează nu este exhaustivă, ci doar o selecţie din numeroasele editoare XML excelente care există:

- XMLwriter – Este unul dintre cele mai bune editoare XML care validează, disponibil ca shareware la `http://xmlwriter.net/`. XMLwriter permite autorului să editeze şi să valideze un document XML faţă de un DTD sau o schemă XML. Autorul poate de asemenea să convertească un document XML în HTML folosind o foaie de stil XSL. Este suportat şi CSS, permiţând astfel formatarea documentelor fie cu XSL fie cu CSS. XMLwriter poate fi folosit şi din linie de comandă pentru a valida şi a converti documente XML.
- XML Spy – Un alt foarte bun editor XML care validează, disponibil ca shareware la `http://www.xmlspy.com/`. XML Spy este un editor XML, XHTML, XSL şi 3DML care foloseşte un mod de vizualizare cu caroiaj a documentului. Deşi îi permite utilizatorului să editeze şi o versiune a fişierului sursă, fişierul sursă efectiv este regenerat la fiecare schimbare a modului de vizualizare, deci formatările se pierd de fiecare dată.
- Amaya – Amaya (`http://www.w3.org/Amaya`) este editorul HTML/XML şi browserul W3C care asigură un suport parţial pentru editarea şi afişarea MathML. Amaya este gratuit şi este printre puţinele editoare HTML/XML care editează şi afişează MathML, limbajul de modelare matematică. În timpul utilizării se poate trece uşor de la modul de editare la cel de navigare, se poate afişa şi edita sursa, structura, legăturile, precum şi alte vederi ale documentului, deci verificarea muncii este uşoară şi puteţi lucra la nivelul cel mai potrivit pentru sarcina pe care trebuie să o rezolvaţi. Amaya este disponibil pentru multe din platformele obişnuite.
- Microsoft XMLNotePad – Este editorul XML simplu al Microsoft. XML Notepad este bine integrat în versiunea XML a Microsoft.
- Emacs – Emacs este probabil cel mai versatil editor pentru programatori, disponibil oriunde în lume, fiind portat pentru toate variantele importante de UNIX, versiunile diferite Windows, Mac OS şi multe altele. Emacs este un motor bun la toate care suportă moduri – module de completare care gestionează diferite limbaje şi sarcini. Modurile disponibile pentru XML sunt psgml, tdtd şi atâtea câte vor mai avea de gând să creeze programatorii.

Browsere cu suport XML

- **InDelv XML Client** – XML Client este un browser cu suport XML/XSL care suportă și CSS pentru XML. Deși este acum într-o variantă beta, pare să fie un produs excelent. O nouă versiune va fi disponibilă pentru descărcare când citiți aceste rânduri. Vizitați situl INDelv la `http://www.indelv.com/`.
- **Microsoft Internet Explorer versiunea 5** – MSIE suportă o versiune XML care parțial nu respectă standardul. Browserul este disponibil oriunde și nu va fi inclus aici, dar merită studiat deoarece, de departe, cele mai multe browsere care sunt active pe Web sunt de la Microsoft. Din nefericire, un procent mare din acestea este format din versiuni vechi, care nu suportă deloc XML.
- **Mozilla** – Proiectul de dezvoltare open-source Mozilla (Netscape) reprezintă o revizuire radicală a poziției Netscape. Militând de multă vreme pentru ideea inovațiilor și extensiilor patentate ca surse de diferențiere a produselor, Netscape s-a convertit în sfârșit la procesul standardizării, iar când Mozilla va deveni versiunea 5 a Netscape – noua generație de browsere – promite să fie browserul cu cele mai multe caracteristici și cel mai apropiat de standarde dintre toate. În locul zurgălăilor din cod, o arie a inovației și diferențierii va fi facilitatea care permite utilizatorului să își definească și să înlocuiască accesoriile, cadrul decorativ și interfața în care este plasată fereastra browserului. Promite de asemenea un suport complet pentru XML, CSS1 și XML, MathML și pentru alte tehnologii bazate pe XML.
- **Panorama SGML Viewer** – Panorama este un browser cu suport pentru SGML și XML care are o lungă istorie în comunitatea SGML. Acum este disponibil ca plug-in care suportă atât browsere Netscape cât și Microsoft de la Interleaf la `http://www.interleaf.com/products/sgml.htm`, împreună cu o suită de produse de creare și publicare SGML înrudite.
- **Plume** – Acest browser Web experimental scris în limbajul de programare Tcl vă permite să vizualizați documente XML sub formă de arbore. Produsul este disponibil de la `http://tcltk.anu.edu.au/`, dar acest sit este deseori indisponibil. Autorul, Steve Ball, lucrează acum la Zveno: `http://www.zveno.com`.
- **Amaya** – Amaya este editorul HTML/XML experimental și browserul W3C care asigură un suport parțial pentru editarea și afișarea MathML (`http://www.w3.org/Amaya`). O descriere mai completă este listată în secțiunea anterioară „Editoare XML".
- **JUMBO** – JUMBO este primul browser XML al lui Peter Murray-Rust și este încă unul dintre foarte puținele browsere care permite vizualizarea fișierelor CML (Chemical Markup Language – limbajul de marcare chimic). Peter Murray-Rust are intenția să dezvolte și să extindă utilizarea CML ca mijloc universal de transfer și de redare pentru informații chimice pe Web. Puteți afla mai multe informații despre acest lucru la `http://www.nottingham.ac.uk/~pazpmr/`.

Instrumente de dezvoltare XML

Iată o colecție de instrumente care încorporează funcții multiple într-un anumit mediu de dezvoltare interactiv. Multe dintre ele suportă dezvoltarea de DTD-uri, foi de stil XSL pentru afișare și alte componente ale unui document XML complet. Următoarele instrumente de dezvoltare sunt bune:

- **Near & Far Designer** – Instrumentul de dezvoltare pentru DTD-uri de la Microstar (achiziționată recent de OpenText Corporation) este disponibil într-un pachet împreună cu editorul XML de la Vervet Logic, XML <PRO>, pentru a forma un mediu de creare DTD XML complet. Îl puteți vedea la `http://www.opentext.com/near_and_far/`.
- **XML <PRO>** – Editorul XML de la Vervet Logic este disponibil într-un pachet împreună cu Near & Far Designer de la Microstar. Vizitați Vervet la `http://www.vervet.com/`.
- **XMLwriter** – Este unul dintre cele mai bune editoare XML care validează, disponibil ca shareware la `http://xmlwriter.net/`.
- **XMetaL 1.2** – Este mai degrabă un produs de creare comercial decât un instrument pentru dezvoltarea de DTD-uri, deși poate fi folosit pentru a dezvolta DTD-uri noi. Aflați mai multe la `http://www.xmetal.com/`.
- **Visual XML** – Un instrument bazat pe Java pentru crearea și editarea de documente XML și de DTD-uri. Visual XML oferă suport pentru handler-ele Java Document, ca și pentru aplicații pure XML, și îl puteți găsi la Bluestone Software la `http://www.bluestone.com/Xml/Visual-XML/`.
- **Python and XML Processing** – Acest sit conține legături spre instrumente Python pentru XML, printre care PyPointers, LT PyXML, tmproc și XML Toolkit. Le puteți vedea la `http://www.python.org/topics/xml/`.
- **XML Spy** – Un alt editor XML care validează, foarte bun, disponibil ca shareware la `http://www.xmlspy.com/`.
- **XML Testbed** – XML Testbed al lui Steve Withall este un mediu general de dezvoltare și testare care include un analizor XML, un groove, un editor și un program care permite vizualizarea arborilor. Îl puteți vedea la `http://www.w3.org/XML/1998/08withall/`.
- **Multidoc Pro Publisher** –. Un browser și un editor XML/SGML de la Citec Information. Citec produce de asemenea Doczilla și The SGML Offshore Toolbox. Le puteți vedea la `http://www.citec.fi/company/products/`.
- **Cocoon** – Cadrul de publicare XML al proiectului Apache. Cocoon include suport pentru Extensible Server Pages (XSP – pagini server extensibile), Xerces-J, Xalan, XT, FOP (generarea PDF din obiectele de formatare XSL) și altele. Puteți găsi mai multe informații la `http://xml.apache.org/`.

Motoare de analizare XML

Un motor de analizare este pur și simplu un instrument care generează un arbore pentru un document XML din acel document și dintr-un DTD opțional. Există multe analizoare care validează, dar acest lucru nu este obligatoriu. Exemplele sunt dintr-o gamă variată, de la instrumente în linie de comandă simple dar eficiente, la medii grafice cu facilități integrate de depanare și editare. Multe pot fi folosite ca părți în seturi de instrumente mai complexe, permițând utilizatorului să evite folosirea unui analizor creat în casă și să se concentreze asupra operațiilor pe care vrea să le facă asupra datelor analizate. Iată câteva motoare de analizare XML:

- **XP** – Un analizor care nu validează de la James Clark. Îl puteți vedea la `http://www.jclark.com/xml/`.
- **SAX** – Simple API for XML. SAX nu este chiar un analizor în sensul obișnuit al

cuvântului ci un motor pentru parcurgerea arborilor XML, bazat pe evenimente, care obține la ieșire modificări ale intrărilor. Poate fi folosit pentru a realiza transformări din XML în HTML și multe alte lucruri utile. SAX este scris în Java, dar are și o implementare Python. Îl puteți vedea la `http://www.megginson. com/SAX/`.

- **Lark** – Lark și Larval sunt analizoarele lui Tim Bray care nu validează, respectiv validează. Le puteți vedea la `http://www.textuality.com/Lark/`.
- **TclXML** – Un analizor care validează scris în Tcl. Îl puteți vedea la `http://www.zveno.com/ zm.cgi/in-tclxml/`.

Instrumente de conversie XML/SGML

Instrumentele de conversie permit transformarea documentelor existente din diferite formate la formatul XML, chiar dacă mai multe sunt listate în secțiunea următoare, „Instrumente XML diverse", care transformă din XML în alte formate cum ar fi PDF sau HTML. Iată trei instrumente de conversie foarte bune:

- **David Raggett's HTML Tidy** – Un excelent instrument în linie de comandă pentru verificarea codului HTML și XHTML pentru erorile cele mai obișnuite. Tidy va încerca de asemenea să corecteze multe din aceste erori, economisind mult din timpul autorului. Tidy poate fi găsit sub multe forme la `http://www.w3.org/people/Raggett/tidy/` și, dacă vi se pare util, vă rog să îi trimiteți lui David o carte poștală interesantă la 73b Ground Corner, Holt, Wiltshire, BA14 6RT, United Kingdom. Nu uitați să lipiți destule timbre dacă trimiteți ilustrata din afara Marii Britanii.
- **HTML-Kit from Chami.com** – HTML-Kit este un editor grafic gratuit pentru Windows care împachetează Tidy, programul de verificare și formatare al lui David Raggett. Ultima versiune (se actualizează rapid) poate fi găsită la `http://www.chami.com/html-kit/`.
- **Microstar FastTAG** – Un program batch și interactiv de conversie, potrivit pentru transformarea aproape oricărui fișier structurat într-unul XML sau în alt tip de fișier structurat, folosind un fișier patentat cu reguli. Regulile pot fi astfel stabilite încât unele părți să fie transformate automat, iar pentru cele dificile să se iasă într-o sesiune interactivă de editare. Microstar a fost achiziționată recent de OpenText Corporation, deci acest produs ar trebui să fie disponibil acum la `http://www.opentext.com/`.

Instrumente XML diverse

Următoarele instrumente sunt îndreptate spre diferitele standarde înrudite cu XML, CSS, XSL/XSLT, RDF, Content Management (gestiunea conținutului) și altele. În această secțiune este inclus și întregul set de instrumente XML al IBM. Deși unele părți ale setului sunt specifice sarcinilor discutate în alte categorii, cum ar fi editarea, validarea XML față de un DTD, editarea DTD și altele, în acest set sunt atât de multe instrumente, încât este mai potrivit să fie prezentat în întregime aici:

- **XT** – Procesorul XSLT al lui James Clark care folosește analizorul XML XP. Îl puteți vedea la `http://www.jclark.com/xml/`.
- **Style One** – Style One este un editor pentru foile de stil CSS1/CSS2 pentru Win98/NT. Îl puteți vedea la `http://www.3-t.com/3-T/products/sytleone/Homepage.html`.
- **4DOM** – Biblioteca FourThought LLC. în Python pentru manipularea documentelor XML folosind modelul documentului obiect de la W3C. O puteți vedea la `http://fourthought.com/ 4Suite/4DOM/`.
- **4XPath** – Biblioteca FourThought LLC. în Python pentru prelucrarea limbajului XPath. O puteți vedea la `http://fourthought.com/4Suite/4XPath/`.
- **4XSLT** – Procesorul de transformare FourThought LLC. în Python pentru o submulțime utilă din XSLT al W3C. Îl puteți vedea la `http://fourthought.com/4Suite/4XSLT/`.
- **Xalan** – Procesoarele pentru foi de stil XSLT ale proiectului Apache, scrise în Java și C++. Le puteți vedea la `http://xml.apache.org/`.
- **FOP** – Transformă obiectele de formatare XSL (XSL Formatting Objects) în fișiere PDF. Îl puteți vedea la `http://xml.apache.org/fop/`[1].
- **Extensibility XML Authority** – Construiește și transformă DTD-uri în SOX și BizTalk. XML Authority va oferi suport și pentru alte dialecte ale XML Schema. Îl puteți vedea la `http://www.extensibility.com/`.
- **Poet Content Management Suite** – Un sistem de baze de date de nivel profesional care permite unei firme mari să își urmărească bunurile informaționale. Îl puteți vedea la `http://www.poet.com/`.
- **BladeRunner** – Un set de gestiune a conținutului bazat pe XML disponibil de la Interleaf la `http://www.interleaf.com/products/`. Setul include X-WAP, XML Wireless Application Product al Interleaf, care folosește standardul XML Wireless Application Protocol (WAP – protocolul aplicațiilor fără fir) și XSL pentru a publica documente XML pentru piața telefoanelor mobile și a agendelor electronice de buzunar cu suport Web.
- **IBM alphaWorks Tool Suite** – O bibliotecă extinsă de instrumente de la laboratoarele de cercetare IBM din întreaga lume. O puteți vedea la `http://www.alphaworks.ibm.com/`. Setul actual de instrumente include:
  - **IBM alphaWorks BML** – Bean Markup Language este un limbaj de script bazat pe XML care folosește Java Beans pentru a alcătui o aplicație.
  - **IBM alphaWorks DDbE** – Data Descriptors by Example (descrieri de date prin exemple) preia un document XML construit corect și construiește un DTD.
  - **IBM alphaWorks DataCraft** – DataCraft asigură o vedere XML asupra bazelor de date și îi permite proiectantului să publice formulare XML pe Web.
  - **IBM alphaWorks DXMLJ** – Dynamic XML for Java îi permite proiectantului să adauge un comportament dinamic la paginile XML folosind adnotări cu zone de nume.
  - **IBM alphaWorks LotusXSL** – O implementare a unei submulțimi a specificației XSL a W3C, în Java.
  - **IBM alphaWorks P3P parser** – O implementare a protocolului P3P (Platform for Privacy Preferences), Java 100% pur.
  - **IBM alphaWorks PatML** – Un procesor pentru identificarea de modele și înlocuire bazat pe reguli pentru XML, în Java Beans, care pot fi încorporate.
  - **IBM alphaWorks SVGView** – Un prototip de program de vizualizare pentru fișiere Scalable Vector Graphics (SVG – grafică vectorială scalabilă), în Java.
  - **IBM alphaWorks TaskGuide Viewer** – Un instrument bazat pe XML pentru crearea și vizualizarea asistenților, ghizi bazați pe cunoștințe care explică sarcini complexe în pași simpli și ușor de urmat.

- **IBM alphaWorks TeXML** – Acest instrument XSLT/Java publică documente XML în TeX – limbajul de formatare și tipărire al lui Donald Knuth – într-un proces format din două părți. Înainte ca transformarea să poată avea loc, trebuie mai întâi să scrieți o foaie de stil de transformare XSLT pentru a converti XML într-un document de tip TeXML. Apoi un instrument Java înglobat, TeXMLatte, realizează analiza inițială într-un DOM TeXML. Un al doilea instrument Java, TeXML, preia un DOM TeXML și îl transformă în Tex.
- **IBM alphaWorks TSpaces** – Un set de instrumente pentru comerț electronic și schimb de date, care suportă XML și alte protocoale. Setul gestionează automat buffere și alte necesități care apar într-un proces de comunicare și stabilește căi de comunicație anonime, asincrone între clienți și servere, fără a cere manipularea unor semafoare sau a altor instrumente de pseudo-simultaneitate.
- **IBM alphaWorks Visual XML Tools** – Un prototip de set de instrumente XML pentru cadrul de aplicație pentru comerț electronic (Application Framework for e-Business). Unul dintre pachetele incluse este Visual XML Query, care vă ajută să construiți o expresie de interogare XML ce vă permite să deschideți un document XML, să construiți o expresie XPath pentru un document XML, să executați expresia XPath folosind motorul XML Query bazat pe XSLT de la Lotus și să salvați expresia XPath într-un fișier. Al doilea pachet este Visual XML Creation, care vă permite să vă conectați la o bază de date DB2 și să obțineți o listă de tabele, să executați o instrucțiune de interogare SQL, să generați un document XML și un DTD și să generați o foaie de stil XSL pentru documentul XML. Al treilea pachet este Visual DTD, un instrument pentru crearea și editarea de DTD-uri. Al patrulea este Visual XML Transformation care vă permite să creați un document XML nou din documente XML existente, folosind XSLT.
- **IBM alphaWorks VoiceXML** – O platformă interactivă XML de recunoaștere a vocii, bazată pe Java, în mare măsură adoptată în domeniul său. VoiceXML poate fi folosită ca bază pentru interacțiunea vocală sau DTMF (TouchTone) cu documentele și datele Web.
- **IBM alphaWorks X-IT** – O aplicație de prelucrare batch pentru fișiere XML care vă permite să prelucrați datele într-un mod interactiv sau non-interactiv.
- **IBM alphaWorks Xeena** – Un editor XML vizual.
- **IBM alphaWorks XMI Toolkit** – Partajează obiecte Java folosind XML, generează DTD-uri și convertește codul între Java, UML și Rational Rose.
- **IBM alphaWorks XML BeanMaker** – Generează un Java Bean dintr-un DTD.
- **IBM alphaWorks XML Diff and Merge Tool** – compară două documente XML, generează o listă de modificări și le acceptă sau le respinge interactiv.
- **IBM alphaWorks XML EditorMaker** – Generează automat un editor vizual personalizat dintr-un DTD XML.
- **IBM alphaWorks XML XLE** – XML Lightweight Extractor permite utilizatorului să marcheze un DTD cu notații patentate și să extragă date dintr-o bază de date, fără a procesa interogări specifice bazei de date.
- **IBM alphaWorks XML XMas** – XML Master generează Java Beans personalizate pentru un anumit document XML. Xmas cuprinde și o colecție de Java Beans utile.
- **IBM alphaWorks XML Enabler** – Un servlet Java care implementează foi de stil LotusXSL pe partea de server. Aceasta permite utilizatorilor cu orice browser să vizualizeze un document XML formatat cu XSL în funcție de browserul pe care îl utilizează. Trebuie să fie definite foi de stil separate pentru fiecare tip de operație de conversie de browser.
- **IBM alphaWorks XML for C++** – Două biblioteci C++ pentru analizarea, validarea și manipularea documentelor XML.
- **IBM alphaWorks XML Generator** – Generează documente XML „aleatoare" dintr-un DTD, pentru a-l testa.
- **IBM alphaWorks XML Parser for Java** – Un analizor XML care validează, pentru Java.
- **IBM alphaWorks XML Productivity Kit for Java** – Un set însoțitor de Java Beans pentru analizorul XML cu validare de mai sus, pentru Java. Folosind setul de productivitate, autorul poate realiza rapid aplicații de comerț electronic simple dar eficiente, introducere de comenzi, coș de cumpărături și alte aplicații XML.
- **IBM alphaWorks XML Security Suite** – O implementare Java a diferitelor caracteristici de securitate cum ar fi semnături digitale, criptarea datelor și controlul accesului.
- **IBM alphaWorks XML Translator Generator** – Această utilitate pentru dezvoltarea de aplicații generează translatori speciali pentru transformarea documentelor XML bazate pe un DTD, în documente XML bazate pe un alt DTD. Deși acest lucru se poate realiza și prin XSLT, aceste instrumente sunt mai comode în utilizare.
- **IBM alphaWorks XML TreeDiff** – Acest pachet de Java Beans vă permite să comparați și să actualizați rapid arbori DOM.
- **IBM alphaWorks XML Viewer** – Acest instrument de afișare Java vă permite să vizualizați datele XML din sursa XML, DTD-ul sau structura ierarhică a documentului.
- **IBM alphaWorks Xplorer** – Un însoțitor al XML Viewer care vă permite să căutați fișiere XML după criterii specifice, să le validați iar apoi să le vedeți cu XML Viewer.

Aplicații cu suport XML

O aplicație cu suport XML este în principal creată pentru a realiza o altă aplicație, cum ar fi așezarea și editarea textului matematic, dar folosește XML ca mecanism de ieșire sau de transfer fie în mod prestabilit, fie ca opțiune.

- **MathType** – Un editor de ecuații matematice pentru XML la `http://www.mathtype.com/`.
- **WebEQ** – Un editor de ecuații matematice pentru XML de la Geometry Technologies, disponibil la `http://www.webeq.com/`.
- **Mathematica** – Mathematica este un produs fantastic dacă sunteți un matematician, un om de știință sau un student care folosește intens matematica. Departe de a fi un simplu editor de ecuații, este un mediu complet de programare pentru matematică, permițându-vă să introduceți, să evaluați și să reprezentați grafic funcții de aproape orice dimensiune și complexitate. De asemenea generează cod MathML, astfel încât

puteți să partajați fișierele cu cercetători care folosesc alte motoare, sau să le folosiți pentru alte scopuri. Îl puteți vedea la `http://www.wolfram.com/`.

- **Amaya** – Editorul HTML/XML și browserul Web experimental al W3C, cu suport și capacități de afișare MathML parțiale. Îl puteți vedea la `http://www.w3.org/Amaya/`.
- **Breeze Commerce Studio** – Breeze Commerce Studio generează cod Java pentru integrarea fișierelor XML și legate de XML, cum ar fi DTD-uri, scheme XML și altele, împreună cu surse de baze de date ODBC/JDBC, în soluții complete de comerț electronic. Îl puteți vedea la `http://www.vsi. com/breeze/`.

Entităţi caracter

ANEXA

C

În această anexă

Seturile de caractere XML și XHTML

Seturile de caractere XML şi XHTML

Spre deosebire de HTML, care se bazează ferm pe convențiile alfabetului vest-european standard, XML și XHTML sunt internaționale ca posibilități și design. Asta înseamnă că, spre deosebire de HTML, nu există seturi de caractere extinse prestabilite pe care să se bazeze dezvoltatorul, în afara setului de caractere de bază ASCII/ISO-IEC 646, care este folosit pentru descrierea limbajului însuși. Orice entități caracter speciale de care aveți nevoie trebuie să fie declarate. Deci un proiectant rus va apela seturile de entități chirilice, un proiectant chinez va alege din numeroasele variante de seturi de caractere chinezești, iar vorbitorii de limbi vest-europene, ca engleza, va trebui să identifice un anumit set extins de caractere la fel ca toți ceilalți utilizatori din lume.

Desigur, dacă în aplicația dumneavoastră nu sunt necesare alte caractere în afară de cele ASCII, nu trebuie să faceți nimic. Și puteți numi explicit aproape orice caracter în orice spațiu de caractere, în mod numeric, dacă este un tab (09), un retur de car (0D), un salt la rând nou (0A) sau este în intervalele 0020-D7FF ori 010000-10FFFF, caracterele permise de Unicode și ISO 10646. Sistemul dumneavoastră de operare și cele ale cititorilor dumneavoastră pot să nu suporte și să nu utilizeze aceste caractere, sau poate va trebui să căutați și să folosiți un anumit suport pentru seturile de caractere extinse, dar capacitatea de a le folosi există în limbaj. Literalmente, un milion sau două de caractere vă stau la dispoziție.

Referirea seturilor de caractere

Referirea și utilizarea seturilor de caractere prin nume mnemonice, și nu în mod numeric, sunt destul de simple și urmează formatul de declarare a entităților folosit pentru substituțiile din XML. Iată cum se apelează ISO Latin-1 și alte câteva seturi de caractere standard, presupunând că le aveți ascunse undeva în sistem sau, în caz contrar, aveți o modalitate oarecare de a le indica:

```
<!ENTITY % ISOnum PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special
          Graphic//EN" "ISOnum.ent">
%ISOnum;
<!ENTITY % ISOpub PUBLIC "ISO 8879:1986//ENTITIES Publishing//EN"
          "ISOpub.ent">
```

```
%ISOpub;
<!ENTITY % ISOtech PUBLIC "ISO 8879:1986//ENTITIES General Technical//EN"
          "ISOtech.ent">
%ISOtech;
<!ENTITY % ISOdia PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN"
          "ISOdia.ent">
%ISOdia;
<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN"
          "ISOlat1.ent">
%ISOlat1;
<!ENTITY % ISOlat2 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 2//EN"
          "ISOlat2.ent">
%ISOlat2;
<!ENTITY % ISOamso PUBLIC "ISO 8879:1986//ENTITIES Added Math Symbols:
          Ordinary//EN" "ISOamso.ent">
%ISOamso;
<!ENTITY % ISOgrk1 PUBLIC "ISO 8879:1986//ENTITIES Greek Letters//EN"
          "ISOgrk1.ent">
%ISOgrk1;
<!ENTITY % ISOgrk3 PUBLIC "ISO 8879:1986//ENTITIES Greek Symbols//EN"
          "ISOgrk3.ent">
%ISOgrk3;
```

Sau iată cum puteți indica o versiune a ISO Latin-1 la distanță, pe situl W3C:

```
<!ENTITY % ISOlat1 PUBLIC "-//W3C//ENTITIES Latin1 for XHTML//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent">
%ISOlat1;
```

Ambele referiri sunt necesare pentru fiecare declarare a unui set de caractere. Prima spune sistemului unde să găsească setul de caractere, în timp ce a doua inserează efectiv caracterele în document pentru a putea fi folosite.

Aceste seturi de entități sunt listate în secțiunea care urmează, „Seturile de caractere standard XHTML".

Seturile de caractere standard XHTML

Deși XML permite folosirea multor seturi de caractere într-un document, redefinirea XHTML include cele trei seturi extinse de entități caracter ale HTML, în mod prestabilit, astfel încât XHTML să se comporte aproape ca și HTML.

Entitățile ISO Latin-1

Notă de copyright ISO

Portions (C) International Organization for Standardization 1986

Permission to copy in any form is granted for use with conforming SGML systems and applications as defined in ISO 8879, provided this notice is included in all copies.

(Permisiunea de copiere în orice formă este acordată pentru folosirea în sistemele și aplicațiile conforme SGML, așa cum sunt definite în ISO 8879, cu condiția includerii acestei note în toate copiile.)

Standardul ASCII nu conține suficiente caractere pentru a permite creatorilor de pagini Web să redea majoritatea limbilor europene cu accentele corecte și cu caracterele lor speciale. Setul de caractere Latin-1 extinde ASCII cu accente, caractere speciale și câteva simboluri tipografice comune care fac posibile afișarea și tipărirea în franceză, germană, daneză, spaniolă și în multe alte limbi. Deși vast, acest set extins de caractere nu rezolvă întregul domeniu al limbilor care există în lume:


```
<!-- Latin-1 characters for XHTML -->

<!-- Character entity set. Typical invocation:
    <!ENTITY % HTMLlat1 PUBLIC
       " -//W3C//ENTITIES Latin 1 for XHTML//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml-lat1.ent">
    %HTMLlat1;
-->

<!ENTITY nbsp   " "> <!-- no-break space = non-breaking space, U+00A0
➥ISOnum -->
<!ENTITY iexcl  "&#161;"> <!-- inverted exclamation mark, U+00A1 ISOnum -->
<!ENTITY cent   "&#162;"> <!-- cent sign, U+00A2 ISOnum -->
<!ENTITY pound  "&#163;"> <!-- pound sign, U+00A3 ISOnum -->
<!ENTITY curren "&#164;"> <!-- currency sign, U+00A4 ISOnum -->
<!ENTITY yen    "&#165;"> <!-- yen sign = yuan sign, U+00A5 ISOnum -->
<!ENTITY brvbar "&#166;"> <!-- broken bar = broken vertical bar, U+00A6
➥ISOnum -->
<!ENTITY sect   "&#167;"> <!-- section sign, U+00A7 ISOnum -->
<!ENTITY uml    "&#168;"> <!-- diaeresis = spacing diaeresis, U+00A8 ISOdia -->
<!ENTITY copy   "&#169;"> <!-- copyright sign, U+00A9 ISOnum -->
<!ENTITY ordf   "&#170;"> <!-- feminine ordinal indicator, U+00AA ISOnum -->
<!ENTITY laquo  "&#171;"> <!-- left-pointing double angle quotation mark = left
➥pointing guillemet, U+00AB ISOnum -->
<!ENTITY not    "&#172;"> <!-- not sign = discretionary hyphen, U+00AC ISOnum -->
<!ENTITY shy    "&#173;"> <!-- soft hyphen = discretionary hyphen, U+00AD ISOnum -->
<!ENTITY reg    "&#174;"> <!-- registered sign = registered trade mark sign,
➥U+00AE ISOnum -->
<!ENTITY macr   "&#175;"> <!-- macron = spacing macron = overline = APL
➥overbar, U+00AF ISOdia -->
<!ENTITY deg    "&#176;"> <!-- degree sign, U+00B0 ISOnum -->
<!ENTITY plusmn "&#177;"> <!-- plus-minus sign = plus-or-minus sign, U+00B1
➥ISOnum -->
<!ENTITY sup2   "&#178;"> <!-- superscript two = superscript digit two =
➥squared, U+00B2 ISOnum -->
<!ENTITY sup3   "&#179;"> <!-- superscript three = superscript digit three =
➥cubed, U+00B3 ISOnum -->
<!ENTITY acute  "&#180;"> <!-- acute accent = spacing acute, U+00B4 ISOdia -->
<!ENTITY micro  "&#181;"> <!-- micro sign, U+00B5 ISOnum -->
<!ENTITY para   "&#182;"> <!-- pilcrow sign = paragraph sign, U+00B6 ISOnum -->
<!ENTITY middot "&#183;"> <!-- middle dot = Georgian comma = Greek middle dot,
➥U+00B7 ISOnum -->
<!ENTITY cedil  "&#184;"> <!-- cedilla = spacing cedilla, U+00B8 ISOdia -->
<!ENTITY sup1   "&#185;"> <!-- superscript one = superscript digit one,
➥U+00B9 ISOnum -->
<!ENTITY ordm   "&#186;"> <!-- masculine ordinal indicator, U+00BA ISOnum -->
<!ENTITY raquo  "&#187;"> <!-- right-pointing double angle quotation mark =
➥right pointing guillemet, U+00BB ISOnum -->
```

```
<!ENTITY frac14 "&#188;"> <!-- vulgar fraction one quarter = fraction one
➥quarter, U+00BC ISOnum -->
<!ENTITY frac12 "&#189;"> <!-- vulgar fraction one half = fraction one half,
➥U+00BD ISOnum -->
<!ENTITY frac34 "&#190;"> <!-- vulgar fraction three quarters = fraction three
➥quarters, U+00BE ISOnum -->
<!ENTITY iquest "&#191;"> <!-- inverted question mark = turned question mark,
➥U+00BF ISOnum -->
<!ENTITY Agrave "&#192;"> <!-- latin capital letter A with grave = latin
➥capital letter A grave, U+00C0 ISOlat1 -->
<!ENTITY Aacute "&#193;"> <!-- latin capital letter A with acute, U+00C1
➥ISOlat1 -->
<!ENTITY Acirc  "&#194;"> <!-- latin capital letter A with circumflex, U+00C2
➥ISOlat1 -->
<!ENTITY Atilde "&#195;"> <!-- latin capital letter A with tilde, U+00C3
➥ISOlat1 -->
<!ENTITY Auml   "&#196;"> <!-- latin capital letter A with diaeresis,
➥U+00C4 ISOlat1 -->
<!ENTITY Aring  "&#197;"> <!-- latin capital letter A with ring above = latin
➥capital letter A ring, U+00C5 ISOlat1 -->
<!ENTITY AElig  "&#198;"> <!-- latin capital letter AE = latin capital
➥ligature AE, U+00C6 ISOlat1 -->
<!ENTITY Ccedil "&#199;"> <!-- latin capital letter C with cedilla,
➥U+00C7 ISOlat1 -->
<!ENTITY Egrave "&#200;"> <!-- latin capital letter E with grave, U+00C8
➥ISOlat1 -->
<!ENTITY Eacute "&#201;"> <!-- latin capital letter E with acute, U+00C9
➥ISOlat1 -->
<!ENTITY Ecirc  "&#202;"> <!-- latin capital letter E with circumflex,
➥U+00CA ISOlat1 -->
<!ENTITY Euml   "&#203;"> <!-- latin capital letter E with diaeresis, U+00CB
➥ISOlat1 -->
<!ENTITY Igrave "&#204;"> <!-- latin capital letter I with grave, U+00CC
➥ISOlat1 -->
<!ENTITY Iacute "&#205;"> <!-- latin capital letter I with acute, U+00CD
➥ISOlat1 -->
<!ENTITY Icirc  "&#206;"> <!-- latin capital letter I with circumflex,
➥U+00CE ISOlat1 -->
<!ENTITY Iuml   "&#207;"> <!-- latin capital letter I with diaeresis, U+00CF
➥ISOlat1 -->
<!ENTITY ETH    "&#208;"> <!-- latin capital letter ETH, U+00D0 ISOlat1 -->
<!ENTITY Ntilde "&#209;"> <!-- latin capital letter N with tilde, U+00D1
➥ISOlat1 -->
<!ENTITY Ograve "&#210;"> <!-- latin capital letter O with grave, U+00D2
➥ISOlat1 -->
<!ENTITY Oacute "&#211;"> <!-- latin capital letter O with acute, U+00D3
➥ISOlat1 -->
<!ENTITY Ocirc  "&#212;"> <!-- latin capital letter O with circumflex, U+00D4
➥ISOlat1 -->
<!ENTITY Otilde "&#213;"> <!-- latin capital letter O with tilde, U+00D5
➥ISOlat1 -->
<!ENTITY Ouml   "&#214;"> <!-- latin capital letter O with diaeresis, U+00D6
➥ISOlat1 -->
```



```
<!ENTITY times  "&#215;"> <!-- multiplication sign, U+00D7 ISOnum -->
<!ENTITY Oslash "&#216;"> <!-- latin capital letter O with stroke = latin
➥capital letter O slash, U+00D8 ISOlat1 -->
<!ENTITY Ugrave "&#217;"> <!-- latin capital letter U with grave, U+00D9
➥ISOlat1 -->
<!ENTITY Uacute "&#218;"> <!-- latin capital letter U with acute, U+00DA
➥ISOlat1 -->
<!ENTITY Ucirc  "&#219;"> <!-- latin capital letter U with circumflex, U+00DB
➥ISOlat1 -->
<!ENTITY Uuml   "&#220;"> <!-- latin capital letter U with diaeresis, U+00DC
➥ISOlat1 -->
<!ENTITY Yacute "&#221;"> <!-- latin capital letter Y with acute, U+00DD
➥ISOlat1 -->
<!ENTITY THORN  "&#222;"> <!-- latin capital letter THORN, U+00DE ISOlat1 -->
<!ENTITY szlig  "&#223;"> <!-- latin small letter sharp s = ess-zed, U+00DF
➥ISOlat1 -->
<!ENTITY agrave "&#224;"> <!-- latin small letter a with grave = latin small
➥letter a grave, U+00E0 ISOlat1 -->
<!ENTITY aacute "&#225;"> <!-- latin small letter a with acute, U+00E1
➥ISOlat1 -->
<!ENTITY acirc  "&#226;"> <!-- latin small letter a with circumflex, U+00E2
➥ISOlat1 -->
<!ENTITY atilde "&#227;"> <!-- latin small letter a with tilde, U+00E3
➥ISOlat1 -->
<!ENTITY auml   "&#228;"> <!-- latin small letter a with diaeresis, U+00E4
➥ISOlat1 -->
<!ENTITY aring  "&#229;"> <!-- latin small letter a with ring above = latin
➥small letter a ring, U+00E5 ISOlat1 -->
<!ENTITY aelig  "&#230;"> <!-- latin small letter ae = latin small ligature
➥ae, U+00E6 ISOlat1 -->
<!ENTITY ccedil "&#231;"> <!-- latin small letter c with cedilla, U+00E7
➥ISOlat1 -->
<!ENTITY egrave "&#232;"> <!-- latin small letter e with grave, U+00E8
➥ISOlat1 -->
<!ENTITY eacute "&#233;"> <!-- latin small letter e with acute, U+00E9
➥ISOlat1 -->
<!ENTITY ecirc  "&#234;"> <!-- latin small letter e with circumflex, U+00EA
➥ISOlat1 -->
<!ENTITY euml   "&#235;"> <!-- latin small letter e with diaeresis, U+00EB
➥ISOlat1 -->
<!ENTITY igrave "&#236;"> <!-- latin small letter i with grave, U+00EC ISOlat1 -->
<!ENTITY iacute "&#237;"> <!-- latin small letter i with acute, U+00ED ISOlat1 -->
<!ENTITY icirc  "&#238;"> <!-- latin small letter i with circumflex, U+00EE
➥ISOlat1 -->
<!ENTITY iuml   "&#239;"> <!-- latin small letter i with diaeresis, U+00EF
➥ISOlat1 -->
<!ENTITY eth    "&#240;"> <!-- latin small letter eth, U+00F0 ISOlat1 -->
<!ENTITY ntilde "&#241;"> <!-- latin small letter n with tilde, U+00F1 ISOlat1 -->
<!ENTITY ograve "&#242;"> <!-- latin small letter o with grave, U+00F2 ISOlat1 -->
<!ENTITY oacute "&#243;"> <!-- latin small letter o with acute, U+00F3 ISOlat1 -->
<!ENTITY ocirc  "&#244;"> <!-- latin small letter o with circumflex, U+00F4
➥ISOlat1 -->
<!ENTITY otilde "&#245;"> <!-- latin small letter o with tilde, U+00F5 ISOlat1 -->
```

```
<!ENTITY ouml   "&#246;"> <!-- latin small letter o with diaeresis, U+00F6
➥ISOlat1 -->
<!ENTITY divide "&#247;"> <!-- division sign, U+00F7 ISOnum -->
<!ENTITY oslash "&#248;"> <!-- latin small letter o with stroke, = latin
➥small letter o slash, U+00F8 ISOlat1 -->
<!ENTITY ugrave "&#249;"> <!-- latin small letter u with grave, U+00F9 ISOlat1 -->
<!ENTITY uacute "&#250;"> <!-- latin small letter u with acute, U+00FA ISOlat1 -->
<!ENTITY ucirc  "&#251;"> <!-- latin small letter u with circumflex, U+00FB
➥ISOlat1 -->
<!ENTITY uuml   "&#252;"> <!-- latin small letter u with diaeresis, U+00FC
➥ISOlat1 -->
<!ENTITY yacute "&#253;"> <!-- latin small letter y with acute, U+00FD ISOlat1 -->
<!ENTITY thorn  "&#254;"> <!-- latin small letter thorn with, U+00FE ISOlat1 -->
<!ENTITY yuml   "&#255;"> <!-- latin small letter y with diaeresis, U+00FF
➥ISOlat1 -->
```

Entităţile caracter speciale ISO

Notă de copyright ISO

Portions (C) International Organization for Standardization 1986

Permission to copy in any form is granted for use with conforming SGML systems and applications as defined in ISO 8879, provided this notice is included in all copies.

(Permisiunea de copiere în orice formă este acordată pentru folosirea în sistemele şi aplicaţiile conforme SGML, aşa cum sunt definite în ISO 8879, cu condiţia includerii acestei note în toate copiile.)

Acest mic set de caractere extinde ASCII şi include caractere speciale folosite în XML şi în standardele înrudite, printre care XHTML, câteva caractere foarte rar folosite în câteva limbi europene şi destul de multe elemente tipografice folosite în formatarea şi marcarea textelor:

```
<!-- Special characters for XHTML -->

<!-- Character entity set. Typical invocation:
     <!ENTITY % HTMLspecial PUBLIC
        "-//W3C//ENTITIES Special for XHTML//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml-special.ent">
     %HTMLspecial;
-->

<!-- Relevant ISO entity set is given unless names are newly introduced.
     New names (i.e., not in ISO 8879 list) do not clash with any
     existing ISO 8879 entity names. ISO 10646 character numbers
     are given for each character, in hex. values are decimal
     conversions of the ISO 10646 values and refer to the document
     character set. Names are Unicode names.
-->

<!-- C0 Controls and Basic Latin -->
<!ENTITY quot     "&#34;"> <!--  quotation mark = APL quote, U+0022 ISOnum -->
<!ENTITY amp      "&#38;"> <!--  ampersand, U+0026 ISOnum -->
<!ENTITY lt       "&#60;"> <!--  less-than sign, U+003C ISOnum -->
```



```
<!ENTITY gt       "&#62;"> <!--  greater-than sign, U+003E ISOnum -->
<!ENTITY apos     "&#39;"> <!--  apostrophe mark, U+0027 ISOnum -->

<!-- Latin Extended-A -->
<!ENTITY OElig    "&#338;"> <!--  latin capital ligature OE, U+0152 ISOlat2 -->
<!ENTITY oelig    "&#339;"> <!--  latin small ligature oe, U+0153 ISOlat2 -->
<!-- ligature is a misnomer, this is a separate character in some languages -->
<!ENTITY Scaron   "&#352;"> <!--  latin capital letter S with caron, U+0160
➥ISOlat2 -->
<!ENTITY scaron   "&#353;"> <!--  latin small letter s with caron, U+0161 ISOlat2 -->
<!ENTITY Yuml     "&#376;"> <!--  latin capital letter Y with diaeresis, U+0178
➥ISOlat2 -->

<!-- Spacing Modifier Letters -->
<!ENTITY circ     "&#710;"> <!--  modifier letter circumflex accent, U+02C6
➥ISOpub -->
<!ENTITY tilde    "&#732;"> <!--  small tilde, U+02DC ISOdia -->

<!-- General Punctuation -->
<!ENTITY ensp     " "> <!-- en space, U+2002 ISOpub -->
<!ENTITY emsp     " "> <!-- em space, U+2003 ISOpub -->
<!ENTITY thinsp   " "> <!-- thin space, U+2009 ISOpub -->
<!ENTITY zwnj     "&#8204;"> <!-- zero width non-joiner, U+200C NEW RFC 2070 -->
<!ENTITY zwj      "&#8205;"> <!-- zero width joiner, U+200D NEW RFC 2070 -->
<!ENTITY lrm      "&#8206;"> <!-- left-to-right mark, U+200E NEW RFC 2070 -->
<!ENTITY rlm      "&#8207;"> <!-- right-to-left mark, U+200F NEW RFC 2070 -->
<!ENTITY ndash    "&#8211;"> <!-- en dash, U+2013 ISOpub -->
<!ENTITY mdash    "&#8212;"> <!-- em dash, U+2014 ISOpub -->
<!ENTITY lsquo    "&#8216;"> <!-- left single quotation mark, U+2018 ISOnum -->
<!ENTITY rsquo    "&#8217;"> <!-- right single quotation mark, U+2019 ISOnum -->
<!ENTITY sbquo    "&#8218;"> <!-- single low-9 quotation mark, U+201A NEW -->
<!ENTITY ldquo    "&#8220;"> <!-- left double quotation mark, U+201C ISOnum -->
<!ENTITY rdquo    "&#8221;"> <!-- right double quotation mark, U+201D ISOnum -->
<!ENTITY bdquo    "&#8222;"> <!-- double low-9 quotation mark, U+201E NEW -->
<!ENTITY dagger   "&#8224;"> <!-- dagger, U+2020 ISOpub -->
<!ENTITY Dagger   "&#8225;"> <!-- double dagger, U+2021 ISOpub -->
<!ENTITY permil   "&#8240;"> <!-- per mille sign, U+2030 ISOtech -->
<!ENTITY lsaquo   "&#8249;"> <!-- single left-pointing angle quotation mark,
➥U+2039 ISO proposed -->
<!-- lsaquo is proposed but not yet ISO standardized -->
<!ENTITY rsaquo   "&#8250;"> <!-- single right-pointing angle quotation mark,
➥U+203A ISO proposed -->
<!-- rsaquo is proposed but not yet ISO standardized -->
<!ENTITY euro     "&#8364;"> <!--  euro sign, U+20AC NEW -->
```

Entităţile caracter matematice, greceşti şi simbolice ISO

A treia lipsă majoră în setul de caractere ASCII, cel puţin din punctul de vedere al europenilor, este lipsa de suport pentru tipografia matematică şi ştiinţifică. Deci acest set de caractere defineşte alfabetul grecesc fără accente, caracterele matematice şi câteva caractere ştiinţifice:

Notă Desigur, acestea nu ajung pentru o matematică serioasă; matematicienii inventează simboluri noi în ritmul rapid al descoperirii unor noi perspective asupra lucrurilor, dar sunt suficiente pentru chestiunile elementare. Iar caracterele greceşti nu se pot folosi pentru tipărirea în greaca modernă.

Notă de copyright ISO

Portions (C) International Organization for Standardization 1986
Permission to copy in any form is granted for use with conforming SGML systems and applications as defined in ISO 8879, provided this notice is included in all copies.
(Permisiunea de copiere în orice formă este acordată pentru folosirea în sistemele şi aplicaţiile conforme SGML, aşa cum sunt definite în ISO 8879, cu condiţia includerii acestei note în toate copiile.)

```
<!-- Mathematical, Greek and Symbolic characters for XHTML -->

<!-- Character entity set. Typical invocation:
     <!ENTITY % HTMLsymbol PUBLIC
        "-//W3C//ENTITIES Symbols for XHTML//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml-symbol.ent">
     %HTMLsymbol;
-->

<!-- Relevant ISO entity set is given unless names are newly introduced.
     New names (i.e., not in ISO 8879 list) do not clash with any
     existing ISO 8879 entity names. ISO 10646 character numbers
     are given for each character, in hex. values are decimal
     conversions of the ISO 10646 values and refer to the document
     conversions of the ISO 10646 values and refer to the document
     character set. Names are Unicode names.
-->

<!-- Latin Extended-B -->
<!ENTITY fnof     "&#402;"> <!-- latin small f with hook = function
                                 = florin, U+0192 ISOtech -->


<!-- Greek -->
<!ENTITY Alpha    "&#913;"> <!-- greek capital letter alpha, U+0391 -->
<!ENTITY Beta     "&#914;"> <!-- greek capital letter beta, U+0392 -->
<!ENTITY Gamma    "&#915;"> <!-- greek capital letter gamma, U+0393 ISOgrk3 -->
<!ENTITY Delta    "&#916;"> <!-- greek capital letter delta, U+0394 ISOgrk3 -->
<!ENTITY Epsilon  "&#917;"> <!-- greek capital letter epsilon, U+0395 -->
<!ENTITY Zeta     "&#918;"> <!-- greek capital letter zeta, U+0396 -->
<!ENTITY Eta      "&#919;"> <!-- greek capital letter eta, U+0397 -->
<!ENTITY Theta    "&#920;"> <!-- greek capital letter theta, U+0398 ISOgrk3 -->
<!ENTITY Iota     "&#921;"> <!-- greek capital letter iota, U+0399 -->
<!ENTITY Kappa    "&#922;"> <!-- greek capital letter kappa, U+039A -->
<!ENTITY Lambda   "&#923;"> <!-- greek capital letter lambda, U+039B ISOgrk3 -->
<!ENTITY Mu       "&#924;"> <!-- greek capital letter mu, U+039C -->
<!ENTITY Nu       "&#925;"> <!-- greek capital letter nu, U+039D -->
<!ENTITY Xi       "&#926;"> <!-- greek capital letter xi, U+039E ISOgrk3 -->
<!ENTITY Omicron  "&#927;"> <!-- greek capital letter omicron, U+039F -->
<!ENTITY Pi       "&#928;"> <!-- greek capital letter pi, U+03A0 ISOgrk3 -->
```



```
<!ENTITY Rho      "&#929;"> <!-- greek capital letter rho, U+03A1 -->
<!-- there is no Sigmaf, and no U+03A2 character either -->
<!ENTITY Sigma    "&#931;"> <!-- greek capital letter sigma, U+03A3 ISOgrk3 -->
<!ENTITY Tau      "&#932;"> <!-- greek capital letter tau, U+03A4 -->
<!ENTITY Upsilon  "&#933;"> <!-- greek capital letter upsilon, U+03A5 ISOgrk3 -->
<!ENTITY Phi      "&#934;"> <!-- greek capital letter phi, U+03A6 ISOgrk3 -->
<!ENTITY Chi      "&#935;"> <!-- greek capital letter chi, U+03A7 -->
<!ENTITY Psi      "&#936;"> <!-- greek capital letter psi, U+03A8 ISOgrk3 -->
<!ENTITY Omega    "&#937;"> <!-- greek capital letter omega, U+03A9 ISOgrk3 -->

<!ENTITY alpha    "&#945;"> <!-- greek small letter alpha, U+03B1 ISOgrk3 -->
<!ENTITY beta     "&#946;"> <!-- greek small letter beta, U+03B2 ISOgrk3 -->
<!ENTITY gamma    "&#947;"> <!-- greek small letter gamma, U+03B3 ISOgrk3 -->
<!ENTITY delta    "&#948;"> <!-- greek small letter delta, U+03B4 ISOgrk3 -->
<!ENTITY epsilon  "&#949;"> <!-- greek small letter epsilon, U+03B5 ISOgrk3 -->
<!ENTITY zeta     "&#950;"> <!-- greek small letter zeta, U+03B6 ISOgrk3 -->
<!ENTITY eta      "&#951;"> <!-- greek small letter eta, U+03B7 ISOgrk3 -->
<!ENTITY theta    "&#952;"> <!-- greek small letter theta, U+03B8 ISOgrk3 -->
<!ENTITY iota     "&#953;"> <!-- greek small letter iota, U+03B9 ISOgrk3 -->
<!ENTITY kappa    "&#954;"> <!-- greek small letter kappa, U+03BA ISOgrk3 -->
<!ENTITY lambda   "&#955;"> <!-- greek small letter lambda, U+03BB ISOgrk3 -->
<!ENTITY mu       "&#956;"> <!-- greek small letter mu, U+03BC ISOgrk3 -->
<!ENTITY nu       "&#957;"> <!-- greek small letter nu, U+03BD ISOgrk3 -->
<!ENTITY xi       "&#958;"> <!-- greek small letter xi, U+03BE ISOgrk3 -->
<!ENTITY omicron  "&#959;"> <!-- greek small letter omicron, U+03BF NEW -->
<!ENTITY pi       "&#960;"> <!-- greek small letter pi, U+03C0 ISOgrk3 -->
<!ENTITY rho      "&#961;"> <!-- greek small letter rho, U+03C1 ISOgrk3 -->
<!ENTITY sigmaf   "&#962;"> <!-- greek small letter final sigma, U+03C2 ISOgrk3 -->
<!ENTITY sigma    "&#963;"> <!-- greek small letter sigma, U+03C3 ISOgrk3 -->
<!ENTITY tau      "&#964;"> <!-- greek small letter tau, U+03C4 ISOgrk3 -->
<!ENTITY upsilon  "&#965;"> <!-- greek small letter upsilon, U+03C5 ISOgrk3 -->
<!ENTITY phi      "&#966;"> <!-- greek small letter phi, U+03C6 ISOgrk3 -->
<!ENTITY chi      "&#967;"> <!-- greek small letter chi, U+03C7 ISOgrk3 -->
<!ENTITY psi      "&#968;"> <!-- greek small letter psi, U+03C8 ISOgrk3 -->
<!ENTITY omega    "&#969;"> <!-- greek small letter omega, U+03C9 ISOgrk3 -->
<!ENTITY thetasym "&#977;"> <!-- greek small letter theta symbol, U+03D1 NEW -->
<!ENTITY upsih    "&#978;"> <!-- greek upsilon with hook symbol, U+03D2 NEW -->
<!ENTITY piv      "&#982;"> <!-- greek pi symbol, U+03D6 ISOgrk3 -->

<!-- General Punctuation -->
<!ENTITY bull     "&#8226;"> <!-- bullet = black small circle, U+2022 ISOpub  -->
<!-- bullet is NOT the same as bullet operator, U+2219 -->
<!ENTITY hellip   "&#8230;"> <!-- horizontal ellipsis = three dot leader,
➥U+2026 ISOpub  -->
<!ENTITY prime    "&#8242;"> <!-- prime = minutes = feet, U+2032 ISOtech -->
<!ENTITY Prime    "&#8243;"> <!-- double prime = seconds = inches, U+2033 ISOtech -->
<!ENTITY oline    "&#8254;"> <!-- overline = spacing overscore, U+203E NEW -->
<!ENTITY frasl    "&#8260;"> <!-- fraction slash, U+2044 NEW -->

<!-- Letterlike Symbols -->
<!ENTITY weierp   "&#8472;"> <!-- script capital P = power set = Weierstrass p,
➥U+2118 ISOamso -->
```

```
<!ENTITY image      "&#8465;"> <!-- blackletter capital I = imaginary part,
➥U+2111 ISOamso -->
<!ENTITY real       "&#8476;"> <!-- blackletter capital R = real part symbol,
➥U+211C ISOamso -->
<!ENTITY trade      "&#8482;"> <!-- trade mark sign, U+2122 ISOnum -->
<!ENTITY alefsym    "&#8501;"> <!-- alef symbol = first transfinite cardinal,
➥U+2135 NEW -->
<!-- alef symbol is NOT the same as hebrew letter alef, U+05D0 although the
➥same glyph could be used to depict both characters -->

<!-- Arrows -->
<!ENTITY larr       "&#8592;"> <!-- leftwards arrow, U+2190 ISOnum -->
<!ENTITY uarr       "&#8593;"> <!-- upwards arrow, U+2191 ISOnum-->
<!ENTITY rarr       "&#8594;"> <!-- rightwards arrow, U+2192 ISOnum -->
<!ENTITY darr       "&#8595;"> <!-- downwards arrow, U+2193 ISOnum -->
<!ENTITY harr       "&#8596;"> <!-- left right arrow, U+2194 ISOamsa -->
<!ENTITY crarr      "&#8629;"> <!-- downwards arrow with corner leftwards =
➥carriage return, U+21B5 NEW -->
<!ENTITY lArr       "&#8656;"> <!-- leftwards double arrow, U+21D0 ISOtech -->
<!-- Unicode does not say that lArr is the same as the 'is implied by' arrow
➥but also does not have any other character for that function. So ? lArr can be
➥used for 'is implied by' as ISOtech suggests -->
<!ENTITY uArr       "&#8657;"> <!-- upwards double arrow, U+21D1 ISOamsa -->
<!ENTITY rArr       "&#8658;"> <!-- rightwards double arrow, U+21D2 ISOtech -->
<!-- Unicode does not say this is the 'implies' character but does not have
➥another character with this function so ? rArr can be used for 'implies' as
➥ISOtech suggests -->
<!ENTITY dArr       "&#8659;"> <!-- downwards double arrow, U+21D3 ISOamsa -->
<!ENTITY hArr       "&#8660;"> <!-- left right double arrow, U+21D4 ISOamsa -->

<!-- Mathematical Operators -->
<!ENTITY forall     "&#8704;"> <!-- for all, U+2200 ISOtech -->
<!ENTITY part       "&#8706;"> <!-- partial differential, U+2202 ISOtech  -->
<!ENTITY exist      "&#8707;"> <!-- there exists, U+2203 ISOtech -->
<!ENTITY empty      "&#8709;"> <!-- empty set = null set = diameter, U+2205
➥ISOamso -->
<!ENTITY nabla      "&#8711;"> <!-- nabla = backward difference, U+2207 ISOtech -->
<!ENTITY isin       "&#8712;"> <!-- element of, U+2208 ISOtech -->
<!ENTITY notin      "&#8713;"> <!-- not an element of, U+2209 ISOtech -->
<!ENTITY ni         "&#8715;"> <!-- contains as member, U+220B ISOtech -->
<!-- should there be a more memorable name than 'ni'? -->
<!ENTITY prod       "&#8719;"> <!-- n-ary product = product sign, U+220F
➥ISOamsb -->
<!-- prod is NOT the same character as U+03A0 'greek capital letter pi' though
➥the same glyph might be used for both -->
<!ENTITY sum        "&#8721;"> <!-- n-ary sumation, U+2211 ISOamsb -->
<!-- sum is NOT the same character as U+03A3 'greek capital letter sigma'
➥though the same glyph might be used for both -->
<!ENTITY minus      "&#8722;"> <!-- minus sign, U+2212 ISOtech -->
<!ENTITY lowast     "&#8727;"> <!-- asterisk operator, U+2217 ISOtech -->
<!ENTITY radic      "&#8730;"> <!-- square root = radical sign, U+221A ISOtech -->
<!ENTITY prop       "&#8733;"> <!-- proportional to, U+221D ISOtech -->
<!ENTITY infin      "&#8734;"> <!-- infinity, U+221E ISOtech -->
```



```
<!ENTITY ang        "&#8736;"> <!-- angle, U+2220 ISOamso -->
<!ENTITY and        "&#8743;"> <!-- logical and = wedge, U+2227 ISOtech -->
<!ENTITY or         "&#8744;"> <!-- logical or = vee, U+2228 ISOtech -->
<!ENTITY cap        "&#8745;"> <!-- intersection = cap, U+2229 ISOtech -->
<!ENTITY cup        "&#8746;"> <!-- union = cup, U+222A ISOtech -->
<!ENTITY int        "&#8747;"> <!-- integral, U+222B ISOtech -->
<!ENTITY there4     "&#8756;"> <!-- therefore, U+2234 ISOtech -->
<!ENTITY sim        "&#8764;"> <!-- tilde operator = varies with = similar
➥to, U+223C ISOtech -->
<!-- tilde operator is NOT the same character as the tilde, U+007E, although
➥the same glyph might be used to represent both  -->
<!ENTITY cong       "&#8773;"> <!-- approximately equal to, U+2245 ISOtech -->
<!ENTITY asymp      "&#8776;"> <!-- almost equal to = asymptotic to, U+2248
➥ISOamsr -->
<!ENTITY ne         "&#8800;"> <!-- not equal to, U+2260 ISOtech -->
<!ENTITY equiv      "&#8801;"> <!-- identical to, U+2261 ISOtech -->
<!ENTITY le         "&#8804;"> <!-- less-than or equal to, U+2264 ISOtech -->
<!ENTITY ge         "&#8805;"> <!-- greater-than or equal to, U+2265 ISOtech -->
<!ENTITY sub        "&#8834;"> <!-- subset of, U+2282 ISOtech -->
<!ENTITY sup        "&#8835;"> <!-- superset of, U+2283 ISOtech -->
<!-- note that nsup, 'not a superset of, U+2283' is not covered by the Symbol
➥font encoding and is not included. Should it be, for symmetry? It is in
➥ISOamsn  -->
<!ENTITY nsub       "&#8836;"> <!-- not a subset of, U+2284 ISOamsn -->
<!ENTITY sube       "&#8838;"> <!-- subset of or equal to, U+2286 ISOtech -->
<!ENTITY supe       "&#8839;"> <!-- superset of or equal to, U+2287 ISOtech -->
<!ENTITY oplus      "&#8853;"> <!-- circled plus = direct sum, U+2295 ISOamsb -->
<!ENTITY otimes     "&#8855;"> <!-- circled times = vector product, U+2297
➥ISOamsb -->
<!ENTITY perp       "&#8869;"> <!-- up tack = orthogonal to = perpendicular,
➥U+22A5 ISOtech -->
<!ENTITY sdot       "&#8901;"> <!-- dot operator, U+22C5 ISOamsb -->
<!-- dot operator is NOT the same character as U+00B7 middle dot -->

<!-- Miscellaneous Technical -->
<!ENTITY lceil      "&#8968;"> <!-- left ceiling = apl upstile, U+2308 ISOamsc  -->
<!ENTITY rceil      "&#8969;"> <!-- right ceiling, U+2309 ISOamsc  -->
<!ENTITY lfloor     "&#8970;"> <!-- left floor = apl downstile, U+230A ISOamsc  -->
<!ENTITY rfloor     "&#8971;"> <!-- right floor, U+230B ISOamsc  -->
<!ENTITY lang       "&#9001;"> <!-- left-pointing angle bracket = bra, U+2329
➥ISOtech -->
<!-- lang is NOT the same character as U+003C 'less than' or U+2039 'single
➥left-pointing angle quotation mark' -->
<!ENTITY rang       "&#9002;"> <!-- right-pointing angle bracket = ket, U+232A
➥ISOtech -->
<!-- rang is NOT the same character as U+003E 'greater than' or U+203A 'single
➥right-pointing angle quotation mark' -->

<!-- Geometric Shapes -->
<!ENTITY loz        "&#9674;"> <!-- lozenge, U+25CA ISOpub -->

<!-- Miscellaneous Symbols -->
<!ENTITY spades     "&#9824;"> <!-- black spade suit, U+2660 ISOpub -->
```

```
<!-- black here seems to mean filled as opposed to hollow -->
<!ENTITY clubs      "&#9827;"> <!-- black club suit = shamrock, U+2663 ISOpub -->
<!ENTITY hearts     "&#9829;"> <!-- black heart suit = valentine, U+2665 ISOpub -->
<!ENTITY diams      "&#9830;"> <!-- black diamond suit, U+2666 ISOpub -->
```

Entităţile XHTML sortate alfabetic

Deoarece poate fi destul de greu să găsiţi sau să înţelegeţi rapid referinţa unui caracter mnemonic, am inclus o listă a celor trei seturi de entităţi XHTML standard, sortată alfabetic, astfel încât să le găsiţi cu uşurinţă:

```
Legendă:
[LS] = Caractere Latin 1
[SS] = Caractere speciale
[XS] = Simboluri

Seturile de simboluri XHTML ordonate alfabetic
Observati ca literele mici apar dupa majuscule

AElig   &#198;   latin capital letter AE = latin capital ligature AE,
➥U+00C6 ISOlat1 [LS]
Aacute  &#193;   latin capital letter A with acute, U+00C1 ISOlat1 [LS]
Acirc   &#194;   latin capital letter A with circumflex, U+00C2 ISOlat1 [LS]
Agrave  &#192;   latin capital letter A with grave = latin capital letter A
➥grave, U+00C0 ISOlat1 [LS]
Alpha   &#913;   greek capital letter alpha, U+0391 [XS]
Aring   &#197;   latin capital letter A with ring above = latin capital
➥letter A ring, U+00C5 ISOlat1 [LS]
Atilde  &#195;   latin capital letter A with tilde, U+00C3 ISOlat1 [LS]
Auml    &#196;   latin capital letter A with diaeresis, U+00C4 ISOlat1 [LS]
Beta    &#914;   greek capital letter beta, U+0392 [XS]
Ccedil  &#199;   latin capital letter C with cedilla, U+00C7 ISOlat1 [LS]
Chi     &#935;   greek capital letter chi, U+03A7 [XS]
Dagger  &#8225;  double dagger, U+2021 ISOpub [SS]
Delta   &#916;   greek capital letter delta, U+0394 ISOgrk3 [XS]
ETH     &#208;   latin capital letter ETH, U+00D0 ISOlat1 [LS]
Eacute  &#201;   latin capital letter E with acute, U+00C9 ISOlat1 [LS]
Ecirc   &#202;   latin capital letter E with circumflex, U+00CA ISOlat1 [LS]
Egrave  &#200;   latin capital letter E with grave, U+00C8 ISOlat1 [LS]
Epsilon &#917;   greek capital letter epsilon, U+0395 [XS]
Eta     &#919;   greek capital letter eta, U+0397 [XS]
Euml    &#203;   latin capital letter E with diaeresis, U+00CB ISOlat1 [LS]
Gamma   &#915;   greek capital letter gamma, U+0393 ISOgrk3 [XS]
Iacute  &#205;   latin capital letter I with acute, U+00CD ISOlat1 [LS]
Icirc   &#206;   latin capital letter I with circumflex, U+00CE ISOlat1 [LS]
Igrave  &#204;   latin capital letter I with grave, U+00CC ISOlat1 [LS]
Iota    &#921;   greek capital letter iota, U+0399 [XS]
Iuml    &#207;   latin capital letter I with diaeresis, U+00CF ISOlat1 [LS]
Kappa   &#922;   greek capital letter kappa, U+039A [XS]
Lambda  &#923;   greek capital letter lambda, U+039B ISOgrk3 [XS]
Mu      &#924;   greek capital letter mu, U+039C [XS]
Ntilde  &#209;   latin capital letter N with tilde, U+00D1 ISOlat1 [LS]
Nu      &#925;   greek capital letter nu, U+039D [XS]
OElig   &#338;   latin capital ligature OE, U+0152 ISOlat2 [SS]
```



```
Oacute  &#211;   latin capital letter O with acute, U+00D3 ISOlat1 [LS]
Ocirc   &#212;   latin capital letter O with circumflex, U+00D4 ISOlat1 [LS]
Ograve  &#210;   latin capital letter O with grave, U+00D2 ISOlat1 [LS]
Omega   &#937;   greek capital letter omega, U+03A9 ISOgrk3 [XS]
Omicron &#927;   greek capital letter omicron, U+039F [XS]
Oslash  &#216;   latin capital letter O with stroke = latin capital letter
➥O slash, U+00D8 ISOlat1 [LS]
Otilde  &#213;   latin capital letter O with tilde, U+00D5 ISOlat1 [LS]
Ouml    &#214;   latin capital letter O with diaeresis, U+00D6 ISOlat1 [LS]
Phi     &#934;   greek capital letter phi, U+03A6 ISOgrk3 [XS]
Pi      &#928;   greek capital letter pi, U+03A0 ISOgrk3 [XS]
Prime   &#8243;  double prime = seconds = inches, U+2033 ISOtech [XS]
Psi     &#936;   greek capital letter psi, U+03A8 ISOgrk3 [XS]
Rho     &#929;   greek capital letter rho, U+03A1 [XS]
Scaron  &#352;   latin capital letter S with caron, U+0160 ISOlat2 [SS]
Sigma   &#931;   greek capital letter sigma, U+03A3 ISOgrk3 [XS]
THORN   &#222;   latin capital letter THORN, U+00DE ISOlat1 [LS]
Tau     &#932;   greek capital letter tau, U+03A4 [XS]
Theta   &#920;   greek capital letter theta, U+0398 ISOgrk3 [XS]
Uacute  &#218;   latin capital letter U with acute, U+00DA ISOlat1 [LS]
Ucirc   &#219;   latin capital letter U with circumflex, U+00DB ISOlat1 [LS]
Ugrave  &#217;   latin capital letter U with grave, U+00D9 ISOlat1 [LS]
Upsilon &#933;   greek capital letter upsilon, U+03A5 ISOgrk3 [XS]
Uuml    &#220;   latin capital letter U with diaeresis, U+00DC ISOlat1 [LS]
Xi      &#926;   greek capital letter xi, U+039E ISOgrk3 [XS]
Yacute  &#221;   latin capital letter Y with acute, U+00DD ISOlat1 [LS]
Yuml    &#376;   latin capital letter Y with diaeresis, U+0178 ISOlat2 [SS]
Zeta    &#918;   greek capital letter zeta, U+0396 [XS]
aacute  &#225;   latin small letter a with acute, U+00E1 ISOlat1 [LS]
acirc   &#226;   latin small letter a with circumflex, U+00E2 ISOlat1 [LS]
acute   &#180;   acute accent = spacing acute, U+00B4 ISOdia [LS]
aelig   &#230;   latin small letter ae = latin small ligature ae, U+00E6
➥ISOlat1 [LS]
agrave  &#224;   latin small letter a with grave = latin small letter a
➥grave, U+00E0 ISOlat1 [LS]
alefsym &#8501;  alef symbol = first transfinite cardinal, U+2135 NEW [XS]
alpha   &#945;   greek small letter alpha, U+03B1 ISOgrk3 [XS]
amp     &#38;    ampersand, U+0026 ISOnum [SS]
and     &#8743;  logical and = wedge, U+2227 ISOtech [XS]
ang     &#8736;  angle, U+2220 ISOamso [XS]
apos    &#39;    apostrophe mark, U+0027 ISOnum [SS]
aring   &#229;   latin small letter a with ring above = latin small letter
➥a ring, U+00E5 ISOlat1 [LS]
asymp   &#8776;  almost equal to = asymptotic to, U+2248 ISOamsr [XS]
atilde  &#227;   latin small letter a with tilde, U+00E3 ISOlat1 [LS]
auml    &#228;   latin small letter a with diaeresis, U+00E4 ISOlat1 [LS]
bdquo   &#8222;  double low-9 quotation mark, U+201E NEW [SS]
beta    &#946;   greek small letter beta, U+03B2 ISOgrk3 [XS]
brvbar  &#166;   broken bar = broken vertical bar, U+00A6 ISOnum [LS]
bull    &#8226;  bullet = black small circle, U+2022 ISOpub [XS]
cap     &#8745;  intersection = cap, U+2229 ISOtech [XS]
ccedil  &#231;   latin small letter c with cedilla, U+00E7 ISOlat1 [LS]
cedil   &#184;   cedilla = spacing cedilla, U+00B8 ISOdia [LS]
```

BIBL. CENTR. UNIV. „M. EMINESCU" IAŞI

```
cent    &#162;   cent sign, U+00A2 ISOnum [LS]
chi     &#967;   greek small letter chi, U+03C7 ISOgrk3 [XS]
circ    &#710;   modifier letter circumflex accent, U+02C6 ISOpub [SS]
clubs   &#9827;  black club suit = shamrock, U+2663 ISOpub [XS]
cong    &#8773;  approximately equal to, U+2245 ISOtech [XS]
copy    &#169;   copyright sign, U+00A9 ISOnum [LS]
crarr   &#8629;  downwards arrow with corner leftwards = carriage return,
➥U+21B5 NEW [XS]
cup     &#8746;  union = cup, U+222A ISOtech [XS]
curren  &#164;   currency sign, U+00A4 ISOnum [LS]
dArr    &#8659;  downwards double arrow, U+21D3 ISOamsa [XS]
dagger  &#8224;  dagger, U+2020 ISOpub [SS]
darr    &#8595;  downwards arrow, U+2193 ISOnum [XS]
deg     &#176;   degree sign, U+00B0 ISOnum [LS]
delta   &#948;   greek small letter delta, U+03B4 ISOgrk3 [XS]
diams   &#9830;  black diamond suit, U+2666 ISOpub [XS]
divide  &#247;   division sign, U+00F7 ISOnum [LS]
eacute  &#233;   latin small letter e with acute, U+00E9 ISOlat1 [LS]
ecirc   &#234;   latin small letter e with circumflex, U+00EA ISOlat1 [LS]
egrave  &#232;   latin small letter e with grave, U+00E8 ISOlat1 [LS]
empty   &#8709;  empty set = null set = diameter, U+2205 ISOamso [XS]
emsp       em space, U+2003 ISOpub [SS]
ensp       en space, U+2002 ISOpub [SS]
epsilon &#949;   greek small letter epsilon, U+03B5 ISOgrk3 [XS]
equiv   &#8801;  identical to, U+2261 ISOtech [XS]
eta     &#951;   greek small letter eta, U+03B7 ISOgrk3 [XS]
eth     &#240;   latin small letter eth, U+00F0 ISOlat1 [LS]
euml    &#235;   latin small letter e with diaeresis, U+00EB ISOlat1 [LS]
euro    &#8364;  euro sign, U+20AC NEW [SS]
exist   &#8707;  there exists, U+2203 ISOtech [XS]
fnof    &#402;   latin small f with hook = function = florin, U+0192
➥ISOtech [XS]
forall  &#8704;  for all, U+2200 ISOtech [XS]
frac12  &#189;   vulgar fraction one half = fraction one half, U+00BD ISOnum [LS]
frac14  &#188;   vulgar fraction one quarter = fraction one quarter, U+00BC
➥ISOnum [LS]
frac34  &#190;   vulgar fraction three quarters = fraction three quarters,
➥U+00BE ISOnum [LS]
frasl   &#8260;  fraction slash, U+2044 NEW [XS]
gamma   &#947;   greek small letter gamma, U+03B3 ISOgrk3 [XS]
ge      &#8805;  greater-than or equal to, U+2265 ISOtech [XS]
gt      &#62;    greater-than sign, U+003E ISOnum [SS]
hArr    &#8660;  left right double arrow, U+21D4 ISOamsa [XS]
harr    &#8596;  left right arrow, U+2194 ISOamsa [XS]
hearts  &#9829;  black heart suit = valentine, U+2665 ISOpub [XS]
hellip  &#8230;  horizontal ellipsis = three dot leader, U+2026 ISOpub [XS]
iacute  &#237;   latin small letter i with acute, U+00ED ISOlat1 [LS]
icirc   &#238;   latin small letter i with circumflex, U+00EE ISOlat1 [LS]
iexcl   &#161;   inverted exclamation mark, U+00A1 ISOnum [LS]
igrave  &#236;   latin small letter i with grave, U+00EC ISOlat1 [LS]
image   &#8465;  blackletter capital I = imaginary part, U+2111 ISOamso [XS]
infin   &#8734;  infinity, U+221E ISOtech [XS]
int     &#8747;  integral, U+222B ISOtech [XS]
```

```
iota    &#953;   greek small letter iota, U+03B9 ISOgrk3 [XS]
iquest  &#191;   inverted question mark = turned question mark, U+00BF ISOnum [LS]
isin    &#8712;  element of, U+2208 ISOtech [XS]
iuml    &#239;   latin small letter i with diaeresis, U+00EF ISOlat1 [LS]
kappa   &#954;   greek small letter kappa, U+03BA ISOgrk3 [XS]
lArr    &#8656;  leftwards double arrow, U+21D0 ISOtech [XS]
lambda  &#955;   greek small letter lambda, U+03BB ISOgrk3 [XS]
lang    &#9001;  left-pointing angle bracket = bra, U+2329 ISOtech [XS]
laquo   &#171;   left-pointing double angle quotation mark = left pointing
➥guillemet, U+00AB ISOnum [LS]
larr    &#8592;  leftwards arrow, U+2190 ISOnum [XS]
lceil   &#8968;  left ceiling = apl upstile, U+2308 ISOamsc [XS]
ldquo   &#8220;  left double quotation mark, U+201C ISOnum [SS]
le      &#8804;  less-than or equal to, U+2264 ISOtech [XS]
lfloor  &#8970;  left floor = apl downstile, U+230A ISOamsc [XS]
lowast  &#8727;  asterisk operator, U+2217 ISOtech [XS]
loz     &#9674;  lozenge, U+25CA ISOpub [XS]
lrm     &#8206;  left-to-right mark, U+200E NEW RFC 2070 [SS]
lsaquo  &#8249;  single left-pointing angle quotation mark, U+2039 ISO proposed [SS]
lsquo   &#8216;  left single quotation mark, U+2018 ISOnum [SS]
lt      &#60;    less-than sign, U+003C ISOnum [SS]
macr    &#175;   macron = spacing macron = overline = APL overbar, U+00AF ISOdia [LS]
mdash   &#8212;  em dash, U+2014 ISOpub [SS]
micro   &#181;   micro sign, U+00B5 ISOnum [LS]
middot  &#183;   middle dot = Georgian comma = Greek middle dot, U+00B7 ISOnum [LS]
minus   &#8722;  minus sign, U+2212 ISOtech [XS]
mu      &#956;   greek small letter mu, U+03BC ISOgrk3 [XS]
nabla   &#8711;  nabla = backward difference, U+2207 ISOtech [XS]
nbsp        no-break space = non-breaking space, U+00A0 ISOnum [LS]
ndash   &#8211;  en dash, U+2013 ISOpub [SS]
ne      &#8800;  not equal to, U+2260 ISOtech [XS]
ni      &#8715;  contains as member, U+220B ISOtech [XS]
not     &#172;   not sign = discretionary hyphen, U+00AC ISOnum [LS]
notin   &#8713;  not an element of, U+2209 ISOtech [XS]
nsub    &#8836;  not a subset of, U+2284 ISOamsn [XS]
ntilde  &#241;   latin small letter n with tilde, U+00F1 ISOlat1 [LS]
nu      &#957;   greek small letter nu, U+03BD ISOgrk3 [XS]
oacute  &#243;   latin small letter o with acute, U+00F3 ISOlat1 [LS]
ocirc   &#244;   latin small letter o with circumflex, U+00F4 ISOlat1 [LS]
oelig   &#339;   latin small ligature oe, U+0153 ISOlat2 [SS]
ograve  &#242;   latin small letter o with grave, U+00F2 ISOlat1 [LS]
oline   &#8254;  overline = spacing overscore, U+203E NEW [XS]
omega   &#969;   greek small letter omega, U+03C9 ISOgrk3 [XS]
omicron &#959;   greek small letter omicron, U+03BF NEW [XS]
oplus   &#8853;  circled plus = direct sum, U+2295 ISOamsb [XS]
or      &#8744;  logical or = vee, U+2228 ISOtech [XS]
ordf    &#170;   feminine ordinal indicator, U+00AA ISOnum [LS]
ordm    &#186;   masculine ordinal indicator, U+00BA ISOnum [LS]
oslash  &#248;   latin small letter o with stroke, = latin small letter
➥o slash, U+00F8 ISOlat1 [LS]
otilde  &#245;   latin small letter o with tilde, U+00F5 ISOlat1 [LS]
otimes  &#8855;  circled times = vector product, U+2297 ISOamsb [XS]
ouml    &#246;   latin small letter o with diaeresis, U+00F6 ISOlat1 [LS]
```

```
para    &#182;   pilcrow sign = paragraph sign, U+00B6 ISOnum [LS]
part    &#8706;  partial differential, U+2202 ISOtech [XS]
permil  &#8240;  per mille sign, U+2030 ISOtech [SS]
perp    &#8869;  up tack = orthogonal to = perpendicular, U+22A5 ISOtech [XS]
phi     &#966;   greek small letter phi, U+03C6 ISOgrk3 [XS]
pi      &#960;   greek small letter pi, U+03C0 ISOgrk3 [XS]
piv     &#982;   greek pi symbol, U+03D6 ISOgrk3 [XS]
plusmn  &#177;   plus-minus sign = plus-or-minus sign, U+00B1 ISOnum [LS]
pound   &#163;   pound sign, U+00A3 ISOnum [LS]
prime   &#8242;  prime = minutes = feet, U+2032 ISOtech [XS]
prod    &#8719;  n-ary product = product sign, U+220F ISOamsb [XS]
prop    &#8733;  proportional to, U+221D ISOtech [XS]
psi     &#968;   greek small letter psi, U+03C8 ISOgrk3 [XS]
quot    &#34;    quotation mark = APL quote, U+0022 ISOnum [SS]
rArr    &#8658;  rightwards double arrow, U+21D2 ISOtech [XS]
radic   &#8730;  square root = radical sign, U+221A ISOtech [XS]
rang    &#9002;  right-pointing angle bracket = ket, U+232A ISOtech [XS]
raquo   &#187;   right-pointing double angle quotation mark = right
➥pointing guillemet, U+00BB ISOnum [LS]
rarr    &#8594;  rightwards arrow, U+2192 ISOnum [XS]
rceil   &#8969;  right ceiling, U+2309 ISOamsc [XS]
rdquo   &#8221;  right double quotation mark, U+201D ISOnum [SS]
real    &#8476;  blackletter capital R = real part symbol, U+211C ISOamso [XS]
reg     &#174;   registered sign = registered trade mark sign, U+00AE ISOnum [LS]
rfloor  &#8971;  right floor, U+230B ISOamsc [XS]
rho     &#961;   greek small letter rho, U+03C1 ISOgrk3 [XS]
rlm     &#8207;  right-to-left mark, U+200F NEW RFC 2070 [SS]
rsaquo  &#8250;  single right-pointing angle quotation mark, U+203A ISO proposed [SS]
rsquo   &#8217;  right single quotation mark, U+2019 ISOnum [SS]
sbquo   &#8218;  single low-9 quotation mark, U+201A NEW [SS]
scaron  &#353;   latin small letter s with caron, U+0161 ISOlat2 [SS]
sdot    &#8901;  dot operator, U+22C5 ISOamsb [XS]
sect    &#167;   section sign, U+00A7 ISOnum [LS]
shy     &#173;   soft hyphen = discretionary hyphen, U+00AD ISOnum [LS]
sigma   &#963;   greek small letter sigma, U+03C3 ISOgrk3 [XS]
sigmaf  &#962;   greek small letter final sigma, U+03C2 ISOgrk3 [XS]
sim     &#8764;  tilde operator = varies with = similar to, U+223C ISOtech [XS]
spades  &#9824;  black spade suit, U+2660 ISOpub [XS]
sub     &#8834;  subset of, U+2282 ISOtech [XS]
sube    &#8838;  subset of or equal to, U+2286 ISOtech [XS]
sum     &#8721;  n-ary sumation, U+2211 ISOamsb [XS]
sup     &#8835;  superset of, U+2283 ISOtech [XS]
sup1    &#185;   superscript one = superscript digit one, U+00B9 ISOnum [LS]
sup2    &#178;   superscript two = superscript digit two = squared, U+00B2
➥ISOnum [LS]
sup3    &#179;   superscript three = superscript digit three = cubed,
➥U+00B3 ISOnum [LS]
supe    &#8839;  superset of or equal to, U+2287 ISOtech [XS]
szlig   &#223;   latin small letter sharp s = ess-zed, U+00DF ISOlat1 [LS]
tau     &#964;   greek small letter tau, U+03C4 ISOgrk3 [XS]
there4  &#8756;  therefore, U+2234 ISOtech [XS]
theta   &#952;   greek small letter theta, U+03B8 ISOgrk3 [XS]
thetasym&#977;   greek small letter theta symbol, U+03D1 NEW [XS]
```



```
thinsp     thin space, U+2009 ISOpub [SS]
thorn   &#254;   latin small letter thorn with, U+00FE ISOlat1 [LS]
tilde   &#732;   small tilde, U+02DC ISOdia [SS]
times   &#215;   multiplication sign, U+00D7 ISOnum [LS]
trade   &#8482;  trade mark sign, U+2122 ISOnum [XS]
uArr    &#8657;  upwards double arrow, U+21D1 ISOamsa [XS]
uacute  &#250;   latin small letter u with acute, U+00FA ISOlat1 [LS]
uarr    &#8593;  upwards arrow, U+2191 ISOnum [XS]
ucirc   &#251;   latin small letter u with circumflex, U+00FB ISOlat1 [LS]
ugrave  &#249;   latin small letter u with grave, U+00F9 ISOlat1 [LS]
uml     &#168;   diaeresis = spacing diaeresis, U+00A8 ISOdia [LS]
upsih   &#978;   greek upsilon with hook symbol, U+03D2 NEW [XS]
upsilon &#965;   greek small letter upsilon, U+03C5 ISOgrk3 [XS]
uuml    &#252;   latin small letter u with diaeresis, U+00FC ISOlat1 [LS]
weierp  &#8472;  script capital P = power set = Weierstrass p, U+2118 ISOamso [XS]
xi      &#958;   greek small letter xi, U+03BE ISOgrk3 [XS]
yacute  &#253;   latin small letter y with acute, U+00FD ISOlat1 [LS]
yen     &#165;   yen sign = yuan sign, U+00A5 ISOnum [LS]
yuml    &#255;   latin small letter y with diaeresis, U+00FF ISOlat1 [LS]
zeta    &#950;   greek small letter zeta, U+03B6 ISOgrk3 [XS]
zwj     &#8205;  zero width joiner, U+200D NEW RFC 2070 [SS]
zwnj    &#8204;  zero width non-joiner, U+200C NEW RFC 2070 [SS]
```

Entităţile XHTML sortate numeric

Deoarece poate fi destul de greu să transformaţi o referinţă numerică la un caracter făcută in-line, am inclus o listă a celor trei seturi de entităţi XHTML standard, sortată numeric, astfel încât să le găsiţi cu uşurinţă:

```
Legendă:
[LS] = Caractere Latin 1
[SS] = Caractere speciale
[XS] = Simboluri

Seturile de simboluri XHTML ordonate numeric

quot    &#34;    quotation mark = APL quote, U+0022 ISOnum [SS]
amp     &#38;    ampersand, U+0026 ISOnum [SS]
apos    &#39;    apostrophe mark, U+0027 ISOnum [SS]
lt      &#60;    less-than sign, U+003C ISOnum [SS]
gt      &#62;    greater-than sign, U+003E ISOnum [SS]
nbsp        no-break space = non-breaking space, U+00A0 ISOnum [LS]
iexcl   &#161;   inverted exclamation mark, U+00A1 ISOnum [LS]
cent    &#162;   cent sign, U+00A2 ISOnum [LS]
pound   &#163;   pound sign, U+00A3 ISOnum [LS]
curren  &#164;   currency sign, U+00A4 ISOnum [LS]
yen     &#165;   yen sign = yuan sign, U+00A5 ISOnum [LS]
brvbar  &#166;   broken bar = broken vertical bar, U+00A6 ISOnum [LS]
sect    &#167;   section sign, U+00A7 ISOnum [LS]
uml     &#168;   diaeresis = spacing diaeresis, U+00A8 ISOdia [LS]
copy    &#169;   copyright sign, U+00A9 ISOnum [LS]
ordf    &#170;   feminine ordinal indicator, U+00AA ISOnum [LS]
laquo   &#171;   left-pointing double angle quotation mark = left pointing
➥guillemet, U+00AB ISOnum [LS]
```

```
not     &#172;   not sign = discretionary hyphen, U+00AC ISOnum [LS]
shy     &#173;   soft hyphen = discretionary hyphen, U+00AD ISOnum [LS]
reg     &#174;   registered sign = registered trade mark sign, U+00AE ISOnum [LS]
macr    &#175;   macron = spacing macron = overline = APL overbar, U+00AF ISOdia [LS]
deg     &#176;   degree sign, U+00B0 ISOnum [LS]
plusmn  &#177;   plus-minus sign = plus-or-minus sign, U+00B1 ISOnum [LS]
sup2    &#178;   superscript two = superscript digit two = squared, U+00B2 ISOnum [LS]
sup3    &#179;   superscript three = superscript digit three = cubed,
↪U+00B3 ISOnum [LS]
acute   &#180;   acute accent = spacing acute, U+00B4 ISOdia [LS]
micro   &#181;   micro sign, U+00B5 ISOnum [LS]
para    &#182;   pilcrow sign = paragraph sign, U+00B6 ISOnum [LS]
middot  &#183;   middle dot = Georgian comma = Greek middle dot, U+00B7 ISOnum [LS]
cedil   &#184;   cedilla = spacing cedilla, U+00B8 ISOdia [LS]
sup1    &#185;   superscript one = superscript digit one, U+00B9 ISOnum [LS]
ordm    &#186;   masculine ordinal indicator, U+00BA ISOnum [LS]
raquo   &#187;   right-pointing double angle quotation mark = right pointing
↪guillemet, U+00BB ISOnum [LS]
frac14  &#188;   vulgar fraction one quarter = fraction one quarter, U+00BC
↪ISOnum [LS]
frac12  &#189;   vulgar fraction one half = fraction one half, U+00BD ISOnum [LS]
frac34  &#190;   vulgar fraction three quarters = fraction three quarters,
↪U+00BE ISOnum [LS]
iquest  &#191;   inverted question mark = turned question mark, U+00BF ISOnum [LS]
Agrave  &#192;   latin capital letter A with grave = latin capital letter A
↪grave, U+00C0 ISOlat1 [LS]
Aacute  &#193;   latin capital letter A with acute, U+00C1 ISOlat1 [LS]
Acirc   &#194;   latin capital letter A with circumflex, U+00C2 ISOlat1 [LS]
Atilde  &#195;   latin capital letter A with tilde, U+00C3 ISOlat1 [LS]
Auml    &#196;   latin capital letter A with diaeresis, U+00C4 ISOlat1 [LS]
Aring   &#197;   latin capital letter A with ring above = latin capital
↪letter A ring, U+00C5 ISOlat1 [LS]
AElig   &#198;   latin capital letter AE = latin capital ligature AE,
↪U+00C6 ISOlat1 [LS]
Ccedil  &#199;   latin capital letter C with cedilla, U+00C7 ISOlat1 [LS]
Egrave  &#200;   latin capital letter E with grave, U+00C8 ISOlat1 [LS]
Eacute  &#201;   latin capital letter E with acute, U+00C9 ISOlat1 [LS]
Ecirc   &#202;   latin capital letter E with circumflex, U+00CA ISOlat1 [LS]
Euml    &#203;   latin capital letter E with diaeresis, U+00CB ISOlat1 [LS]
Igrave  &#204;   latin capital letter I with grave, U+00CC ISOlat1 [LS]
Iacute  &#205;   latin capital letter I with acute, U+00CD ISOlat1 [LS]
Icirc   &#206;   latin capital letter I with circumflex, U+00CE ISOlat1 [LS]
Iuml    &#207;   latin capital letter I with diaeresis, U+00CF ISOlat1 [LS]
ETH     &#208;   latin capital letter ETH, U+00D0 ISOlat1 [LS]
Ntilde  &#209;   latin capital letter N with tilde, U+00D1 ISOlat1 [LS]
Ograve  &#210;   latin capital letter O with grave, U+00D2 ISOlat1 [LS]
Oacute  &#211;   latin capital letter O with acute, U+00D3 ISOlat1 [LS]
Ocirc   &#212;   latin capital letter O with circumflex, U+00D4 ISOlat1 [LS]
Otilde  &#213;   latin capital letter O with tilde, U+00D5 ISOlat1 [LS]
Ouml    &#214;   latin capital letter O with diaeresis, U+00D6 ISOlat1 [LS]
times   &#215;   multiplication sign, U+00D7 ISOnum [LS]
Oslash  &#216;   latin capital letter O with stroke = latin capital letter
↪O slash, U+00D8 ISOlat1 [LS]
Ugrave  &#217;   latin capital letter U with grave, U+00D9 ISOlat1 [LS]
Uacute  &#218;   latin capital letter U with acute, U+00DA ISOlat1 [LS]
Ucirc   &#219;   latin capital letter U with circumflex, U+00DB ISOlat1 [LS]
Uuml    &#220;   latin capital letter U with diaeresis, U+00DC ISOlat1 [LS]
Yacute  &#221;   latin capital letter Y with acute, U+00DD ISOlat1 [LS]
THORN   &#222;   latin capital letter THORN, U+00DE ISOlat1 [LS]
szlig   &#223;   latin small letter sharp s = ess-zed, U+00DF ISOlat1 [LS]
agrave  &#224;   latin small letter a with grave = latin small letter a
↪grave, U+00E0 ISOlat1 [LS]
aacute  &#225;   latin small letter a with acute, U+00E1 ISOlat1 [LS]
acirc   &#226;   latin small letter a with circumflex, U+00E2 ISOlat1 [LS]
atilde  &#227;   latin small letter a with tilde, U+00E3 ISOlat1 [LS]
auml    &#228;   latin small letter a with diaeresis, U+00E4 ISOlat1 [LS]
aring   &#229;   latin small letter a with ring above = latin small letter
↪a ring, U+00E5 ISOlat1 [LS]
aelig   &#230;   latin small letter ae = latin small ligature ae, U+00E6
↪ISOlat1 [LS]
ccedil  &#231;   latin small letter c with cedilla, U+00E7 ISOlat1 [LS]
egrave  &#232;   latin small letter e with grave, U+00E8 ISOlat1 [LS]
eacute  &#233;   latin small letter e with acute, U+00E9 ISOlat1 [LS]
ecirc   &#234;   latin small letter e with circumflex, U+00EA ISOlat1 [LS]
euml    &#235;   latin small letter e with diaeresis, U+00EB ISOlat1 [LS]
igrave  &#236;   latin small letter i with grave, U+00EC ISOlat1 [LS]
iacute  &#237;   latin small letter i with acute, U+00ED ISOlat1 [LS]
icirc   &#238;   latin small letter i with circumflex, U+00EE ISOlat1 [LS]
iuml    &#239;   latin small letter i with diaeresis, U+00EF ISOlat1 [LS]
eth     &#240;   latin small letter eth, U+00F0 ISOlat1 [LS]
ntilde  &#241;   latin small letter n with tilde, U+00F1 ISOlat1 [LS]
ograve  &#242;   latin small letter o with grave, U+00F2 ISOlat1 [LS]
oacute  &#243;   latin small letter o with acute, U+00F3 ISOlat1 [LS]
ocirc   &#244;   latin small letter o with circumflex, U+00F4 ISOlat1 [LS]
otilde  &#245;   latin small letter o with tilde, U+00F5 ISOlat1 [LS]
ouml    &#246;   latin small letter o with diaeresis, U+00F6 ISOlat1 [LS]
divide  &#247;   division sign, U+00F7 ISOnum [LS]
oslash  &#248;   latin small letter o with stroke, = latin small letter
↪o slash, U+00F8 ISOlat1 [LS]
ugrave  &#249;   latin small letter u with grave, U+00F9 ISOlat1 [LS]
uacute  &#250;   latin small letter u with acute, U+00FA ISOlat1 [LS]
ucirc   &#251;   latin small letter u with circumflex, U+00FB ISOlat1 [LS]
uuml    &#252;   latin small letter u with diaeresis, U+00FC ISOlat1 [LS]
yacute  &#253;   latin small letter y with acute, U+00FD ISOlat1 [LS]
thorn   &#254;   latin small letter thorn with, U+00FE ISOlat1 [LS]
yuml    &#255;   latin small letter y with diaeresis, U+00FF ISOlat1 [LS]
OElig   &#338;   latin capital ligature OE, U+0152 ISOlat2 [SS]
oelig   &#339;   latin small ligature oe, U+0153 ISOlat2 [SS]
Scaron  &#352;   latin capital letter S with caron, U+0160 ISOlat2 [SS]
scaron  &#353;   latin small letter s with caron, U+0161 ISOlat2 [SS]
Yuml    &#376;   latin capital letter Y with diaeresis, U+0178 ISOlat2 [SS]
fnof    &#402;   latin small f with hook = function = florin, U+0192
↪ISOtech [XS]
circ    &#710;   modifier letter circumflex accent, U+02C6 ISOpub [SS]
tilde   &#732;   small tilde, U+02DC ISOdia [SS]
Alpha   &#913;   greek capital letter alpha, U+0391 [XS]
```

```
Beta    &#914;   greek capital letter beta, U+0392 [XS]
Gamma   &#915;   greek capital letter gamma, U+0393 ISOgrk3 [XS]
Delta   &#916;   greek capital letter delta, U+0394 ISOgrk3 [XS]
Epsilon &#917;   greek capital letter epsilon, U+0395 [XS]
Zeta    &#918;   greek capital letter zeta, U+0396 [XS]
Eta     &#919;   greek capital letter eta, U+0397 [XS]
Theta   &#920;   greek capital letter theta, U+0398 ISOgrk3 [XS]
Iota    &#921;   greek capital letter iota, U+0399 [XS]
Kappa   &#922;   greek capital letter kappa, U+039A [XS]
Lambda  &#923;   greek capital letter lambda, U+039B ISOgrk3 [XS]
Mu      &#924;   greek capital letter mu, U+039C [XS]
Nu      &#925;   greek capital letter nu, U+039D [XS]
Xi      &#926;   greek capital letter xi, U+039E ISOgrk3 [XS]
Omicron &#927;   greek capital letter omicron, U+039F [XS]
Pi      &#928;   greek capital letter pi, U+03A0 ISOgrk3 [XS]
Rho     &#929;   greek capital letter rho, U+03A1 [XS]
Sigma   &#931;   greek capital letter sigma, U+03A3 ISOgrk3 [XS]
Tau     &#932;   greek capital letter tau, U+03A4 [XS]
Upsilon &#933;   greek capital letter upsilon, U+03A5 ISOgrk3 [XS]
Phi     &#934;   greek capital letter phi, U+03A6 ISOgrk3 [XS]
Chi     &#935;   greek capital letter chi, U+03A7 [XS]
Psi     &#936;   greek capital letter psi, U+03A8 ISOgrk3 [XS]
Omega   &#937;   greek capital letter omega, U+03A9 ISOgrk3 [XS]
alpha   &#945;   greek small letter alpha, U+03B1 ISOgrk3 [XS]
beta    &#946;   greek small letter beta, U+03B2 ISOgrk3 [XS]
gamma   &#947;   greek small letter gamma, U+03B3 ISOgrk3 [XS]
delta   &#948;   greek small letter delta, U+03B4 ISOgrk3 [XS]
epsilon &#949;   greek small letter epsilon, U+03B5 ISOgrk3 [XS]
zeta    &#950;   greek small letter zeta, U+03B6 ISOgrk3 [XS]
eta     &#951;   greek small letter eta, U+03B7 ISOgrk3 [XS]
theta   &#952;   greek small letter theta, U+03B8 ISOgrk3 [XS]
iota    &#953;   greek small letter iota, U+03B9 ISOgrk3 [XS]
kappa   &#954;   greek small letter kappa, U+03BA ISOgrk3 [XS]
lambda  &#955;   greek small letter lambda, U+03BB ISOgrk3 [XS]
mu      &#956;   greek small letter mu, U+03BC ISOgrk3 [XS]
nu      &#957;   greek small letter nu, U+03BD ISOgrk3 [XS]
xi      &#958;   greek small letter xi, U+03BE ISOgrk3 [XS]
omicron &#959;   greek small letter omicron, U+03BF NEW [XS]
pi      &#960;   greek small letter pi, U+03C0 ISOgrk3 [XS]
rho     &#961;   greek small letter rho, U+03C1 ISOgrk3 [XS]
sigmaf  &#962;   greek small letter final sigma, U+03C2 ISOgrk3 [XS]
sigma   &#963;   greek small letter sigma, U+03C3 ISOgrk3 [XS]
tau     &#964;   greek small letter tau, U+03C4 ISOgrk3 [XS]
upsilon &#965;   greek small letter upsilon, U+03C5 ISOgrk3 [XS]
phi     &#966;   greek small letter phi, U+03C6 ISOgrk3 [XS]
chi     &#967;   greek small letter chi, U+03C7 ISOgrk3 [XS]
psi     &#968;   greek small letter psi, U+03C8 ISOgrk3 [XS]
omega   &#969;   greek small letter omega, U+03C9 ISOgrk3 [XS]
thetasym &#977;  greek small letter theta symbol, U+03D1 NEW [XS]
upsih   &#978;   greek upsilon with hook symbol, U+03D2 NEW [XS]
piv     &#982;   greek pi symbol, U+03D6 ISOgrk3 [XS]
ensp       en space, U+2002 ISOpub [SS]
emsp       em space, U+2003 ISOpub [SS]
thinsp     thin space, U+2009 ISOpub [SS]
zwnj    &#8204;  zero width non-joiner, U+200C NEW RFC 2070 [SS]
zwj     &#8205;  zero width joiner, U+200D NEW RFC 2070 [SS]
lrm     &#8206;  left-to-right mark, U+200E NEW RFC 2070 [SS]
rlm     &#8207;  right-to-left mark, U+200F NEW RFC 2070 [SS]
ndash   &#8211;  en dash, U+2013 ISOpub [SS]
mdash   &#8212;  em dash, U+2014 ISOpub [SS]
lsquo   &#8216;  left single quotation mark, U+2018 ISOnum [SS]
rsquo   &#8217;  right single quotation mark, U+2019 ISOnum [SS]
sbquo   &#8218;  single low-9 quotation mark, U+201A NEW [SS]
ldquo   &#8220;  left double quotation mark, U+201C ISOnum [SS]
rdquo   &#8221;  right double quotation mark, U+201D ISOnum [SS]
bdquo   &#8222;  double low-9 quotation mark, U+201E NEW [SS]
dagger  &#8224;  dagger, U+2020 ISOpub [SS]
Dagger  &#8225;  double dagger, U+2021 ISOpub [SS]
bull    &#8226;  bullet = black small circle, U+2022 ISOpub [XS]
hellip  &#8230;  horizontal ellipsis = three dot leader, U+2026 ISOpub [XS]
permil  &#8240;  per mille sign, U+2030 ISOtech [SS]
prime   &#8242;  prime = minutes = feet, U+2032 ISOtech [XS]
Prime   &#8243;  double prime = seconds = inches, U+2033 ISOtech [XS]
lsaquo  &#8249;  single left-pointing angle quotation mark, U+2039 ISO proposed [SS]
rsaquo  &#8250;  single right-pointing angle quotation mark, U+203A ISO proposed [SS]
oline   &#8254;  overline = spacing overscore, U+203E NEW [XS]
frasl   &#8260;  fraction slash, U+2044 NEW [XS]
euro    &#8364;  euro sign, U+20AC NEW [SS]
image   &#8465;  blackletter capital I = imaginary part, U+2111 ISOamso [XS]
weierp  &#8472;  script capital P = power set = Weierstrass p, U+2118 ISOamso [XS]
real    &#8476;  blackletter capital R = real part symbol, U+211C ISOamso [XS]
trade   &#8482;  trade mark sign, U+2122 ISOnum [XS]
alefsym &#8501;  alef symbol = first transfinite cardinal, U+2135 NEW [XS]
larr    &#8592;  leftwards arrow, U+2190 ISOnum [XS]
uarr    &#8593;  upwards arrow, U+2191 ISOnum [XS]
rarr    &#8594;  rightwards arrow, U+2192 ISOnum [XS]
darr    &#8595;  downwards arrow, U+2193 ISOnum [XS]
harr    &#8596;  left right arrow, U+2194 ISOamsa [XS]
crarr   &#8629;  downwards arrow with corner leftwards = carriage return,
➥U+21B5 NEW [XS]
lArr    &#8656;  leftwards double arrow, U+21D0 ISOtech [XS]
uArr    &#8657;  upwards double arrow, U+21D1 ISOamsa [XS]
rArr    &#8658;  rightwards double arrow, U+21D2 ISOtech [XS]
dArr    &#8659;  downwards double arrow, U+21D3 ISOamsa [XS]
hArr    &#8660;  left right double arrow, U+21D4 ISOamsa [XS]
forall  &#8704;  for all, U+2200 ISOtech [XS]
part    &#8706;  partial differential, U+2202 ISOtech [XS]
exist   &#8707;  there exists, U+2203 ISOtech [XS]
empty   &#8709;  empty set = null set = diameter, U+2205 ISOamso [XS]
nabla   &#8711;  nabla = backward difference, U+2207 ISOtech [XS]
isin    &#8712;  element of, U+2208 ISOtech [XS]
notin   &#8713;  not an element of, U+2209 ISOtech [XS]
ni      &#8715;  contains as member, U+220B ISOtech [XS]
prod    &#8719;  n-ary product = product sign, U+220F ISOamsb [XS]
sum     &#8721;  n-ary sumation, U+2211 ISOamsb [XS]
minus   &#8722;  minus sign, U+2212 ISOtech [XS]
```

```
lowast  &#8727;  asterisk operator, U+2217 ISOtech [XS]
radic   &#8730;  square root = radical sign, U+221A ISOtech [XS]
prop    &#8733;  proportional to, U+221D ISOtech [XS]
infin   &#8734;  infinity, U+221E ISOtech [XS]
ang     &#8736;  angle, U+2220 ISOamso [XS]
and     &#8743;  logical and = wedge, U+2227 ISOtech [XS]
or      &#8744;  logical or = vee, U+2228 ISOtech [XS]
cap     &#8745;  intersection = cap, U+2229 ISOtech [XS]
cup     &#8746;  union = cup, U+222A ISOtech [XS]
int     &#8747;  integral, U+222B ISOtech [XS]
there4  &#8756;  therefore, U+2234 ISOtech [XS]
sim     &#8764;  tilde operator = varies with = similar to, U+223C ISOtech [XS]
cong    &#8773;  approximately equal to, U+2245 ISOtech [XS]
asymp   &#8776;  almost equal to = asymptotic to, U+2248 ISOamsr [XS]
ne      &#8800;  not equal to, U+2260 ISOtech [XS]
equiv   &#8801;  identical to, U+2261 ISOtech [XS]
le      &#8804;  less-than or equal to, U+2264 ISOtech [XS]
ge      &#8805;  greater-than or equal to, U+2265 ISOtech [XS]
sub     &#8834;  subset of, U+2282 ISOtech [XS]
sup     &#8835;  superset of, U+2283 ISOtech [XS]
nsub    &#8836;  not a subset of, U+2284 ISOamsn [XS]
sube    &#8838;  subset of or equal to, U+2286 ISOtech [XS]
supe    &#8839;  superset of or equal to, U+2287 ISOtech [XS]
oplus   &#8853;  circled plus = direct sum, U+2295 ISOamsb [XS]
otimes  &#8855;  circled times = vector product, U+2297 ISOamsb [XS]
perp    &#8869;  up tack = orthogonal to = perpendicular, U+22A5 ISOtech [XS]
sdot    &#8901;  dot operator, U+22C5 ISOamsb [XS]
lceil   &#8968;  left ceiling = apl upstile, U+2308 ISOamsc [XS]
rceil   &#8969;  right ceiling, U+2309 ISOamsc [XS]
lfloor  &#8970;  left floor = apl downstile, U+230A ISOamsc [XS]
rfloor  &#8971;  right floor, U+230B ISOamsc [XS]
lang    &#9001;  left-pointing angle bracket = bra, U+2329 ISOtech [XS]
rang    &#9002;  right-pointing angle bracket = ket, U+232A ISOtech [XS]
loz     &#9674;  lozenge, U+25CA ISOpub [XS]
spades  &#9824;  black spade suit, U+2660 ISOpub [XS]
clubs   &#9827;  black club suit = shamrock, U+2663 ISOpub [XS]
hearts  &#9829;  black heart suit = valentine, U+2665 ISOpub [XS]
diams   &#9830;  black diamond suit, U+2666 ISOpub [XS]
```

Referinţe CSS1, CSS2

ANEXA

D

În această anexă

Examinarea unui exemplu de foaie de stil
Referinţă rapidă CSS
Text[S2]
Importanţa CSS
Suportul CSS în browsere
Rezolvarea elegantă a bug-urilor
Referinţe
Note asupra CSS2, generaţia următoare a foilor de stil

Examinarea unui exemplu de foaie de stil

CSS2 este compatibil cu CSS1, cu câteva excepţii minore sau logice, deşi adaugă destul de multe funcţionalităţi. Toate foile de stil respectă în mare acelaşi model, deci iată un exemplu care vă arată cum puteţi să vă setaţi propria pagină cu o foaie de stil inline şi cum să suprascrieţi unele valori prestabilite cu ajutorul atributului `style` dintr-o etichetă.

```
<html>
  <head>
    <style type="text/css">
      <--
      body { margin-left: 3%; margin-right: 3%; color: black;
             font-family: serif; background-color: #def ;
             background: url(background.jpg) ; }
      a:link { color: blue }
      a:visited { color: purple }
      a:active { color: red }
      div.block { margin-left: 9%; }
      div.block H2, DIV.block H3 { margin-left: -9%; }
      h1 { clear: left; margin-top: 2em; text-align: center; }
      pre { font-family: monospace; }
      img { border: 0; }
      -->
    </style>
  </head>
  <body text="black" bgcolor="#ddeeff" background="background.jpg"
```

```
      link="blue" vlink="purple" alink="red">
  <h1 style="color:red background-color:yellow">Main Heading</H1>
```

Această foaie de stil stabileşte marginile standard la 3% din fereastra browserului, culoarea textului la negru, alege un font cu serife fără să specifice unul anume şi stabileşte culoarea fundalului la `#ddeeff` folosind o notaţie prescurtată cu trei cifre hexazecimale, `#def`. Fiecare cifră este dublată pentru a alcătui numărul final `#ddeeff`. De asemenea, încarcă o imagine în fundal după ce culoarea a fost stabilită la o valoare potrivită, astfel încât tranziţia de la pagina albă la fundal să fie lină. Numeroasele perechi proprietate/valoare sunt separate prin punct şi virgulă astfel încât browserul să le poată păstra corect în micul său creier.

Sugestie de la LeeAnne.com Words to weave by

W3C are un serviciu de validare CSS2 excelent, disponibil la `http://jigsaw.w3.org/css-validator/`, şi întotdeauna este o idee bună să validaţi o foaie de stil înainte de utilizare. Puteţi folosi de asemenea serviciul lor de validare HTML/XHTML la `http://validator.w3.org/` înainte de a pierde prea mult timp scărpinându-vă în cap când lucrurile nu merg cum trebuie. Întotdeauna este enervant când sursa unei probleme este o simplă eroare de codare, iar folosirea instrumentelor de validare este o bună practică de proiectare şi codare.

Observaţi că am adus aceleaşi argumente şi în eticheta `BODY`. Deoarece implementarea foilor de stil este în cel mai bun caz neuniformă, ca regulă generală trebuie să daţi argumente şi pentru stilurile vechi şi pentru cele noi în timpul tranziţiei de la o versiune la alta, deşi mecanismul mai vechi este acum criticat.

Modificările marginilor pot să apară sau nu, în funcţie de browser. Marginile din dreapta sunt de obicei mai rar implementate decât cele din stânga, dar pagina va putea fi citită şi fără informaţiile de margine, deci le neglijăm. Dacă apar, suntem fericiţi. Dacă nu, trecem mai departe.

În rest, comenzile de stil sunt rafinamente pe care dorim să le vedem, deci unele din ele vor fi date ca argumente şi în corp, ca poziţie de rezervă. Puteţi vedea ce efect are fiecare căutând cuvintele cheie în lista de referinţe de mai jos.

Referinţă rapidă CSS

Numerele care se găsesc la sfârşitul fiecărei descrieri de proprietate CSS se referă la titlurile secţiunilor efective din recomandările World Wide Web Consortium (W3C) pentru foi de stil, Cascading Style Sheets, Level 1 (CSS1) şi Cascading Style Sheets, Level 2 (CSS2). Unele numere sunt omise deoarece se referă la aspecte care nu au proprietăţi. Numerele de secţiune CSS2 sunt scrise cursiv, fie deasupra proprietăţii, fie deasupra grupului de proprietăţi descrise în secţiunea CSS2, deşi pot să urmeze uneori ordinea CSS1. În toate cazurile, Recomandările CSS1 şi CSS2 efective ale W3C sunt primordiale. Acest rezumat concis este doar un instrument comod pentru o consultare rapidă, nu un manual de studiu.

Fiecare intrare este alcătuită dintr-un selector, de obicei un element HTML urmat de o pereche proprietate/valoare încadrată de acolade, urmat de un exemplu opţional, o descriere sau o explicaţie. Valoarea poate fi o listă de valori pe care le poate lua proprietatea.

Am omis exemplele pentru aproape toate proprietăţile de după „font-family" deoarece sunt toate aceleaşi şi nu numai că ar fi plictisitoare, dar ar ocupa şi un spaţiu care poate fi folosit mai bine. Elementele iniţiale (mai complexe) au toate exemple care să vă ajute să vă daţi seama ce se întâmplă.

Am pus puncte de suspensie (...) după perechile proprietate/valoare pentru a vă arăta că puteţi adăuga mai multe proprietăţi dacă doriţi, dar ţineţi minte că trebuie să separaţi perechile prin punct şi virgulă, aşa cum se arată în primele exemple.

O explicaţie tehnică a CSS1 şi CSS2

Textele recomandărilor CSS1 şi CSS2 sunt surprinzător de lizibile, deci dacă doriţi o explicaţie mai tehnică şi nu vă deranjează să citiţi documente pe calculator, le puteţi căuta pe Web în situl W3C, la `http://www.w3.org/Style/css/`.

W3C are de asemenea îndrumare on-line şi informaţii de bază pentru a uşura viaţa unui viitor *maven* al foilor de stil. (Adică *guru*, pentru cei care preferă să folosească metaforele din Extremul Orient, şi nu pe cele din Europa Centrală.)

Convenţiile folosite în această anexă sunt prezentate în tabelul D.1.

Tabelul D.1 Convenţii de formatare a textului folosite în această anexă

Convenţii de formatare a textului	Semnificaţie
CSS1 x.x	Se referă la paragrafe din Recomandarea CSS1 a W3C
CSS2 x.x	Se referă la paragrafe din Recomandarea CSS2 a W3C
Aldine	Proprietăţi de stil şi valori sigure pentru toate browserele
Normale	Proprietăţi de stil şi valori sigure care nu sunt suportate de toate browserele
Cursive	Comentarii şi explicaţii – şi, de asemenea, titluri de secţiuni CSS2
[adv]	Marchează caracteristici avansate care nu sunt necesare pentru a pretinde conformitatea cu CSS şi nu sunt suportate de majoritatea browserelor
[s2]	Marchează caracteristici introduse în CSS2
[ns2]	Marchează caracteristici al căror comportament a fost modificat şi a căror clasificare a fost modificată în CSS2
`<element>`	Se referă la elemente (etichete) HTML
classid	Se referă la o valoare introdusă într-un atribut de clasă în eticheta HTML
id	Se referă la o valoare introdusă într-un atribut ID în eticheta HTML
Paranteze unghiulare <>	Încadrează elementele HTML, dar sunt folosite şi pentru a încadra o valoare de proprietate pentru a indica faptul că întregul element, inclusiv parantezele, trebuie să fie înlocuit de un singur cuvânt

Acolade {}	Încadrează perechile proprietate/valoare și sunt obligatorii
property	Se referă la proprietățile CSS care se află înainte de semnul două puncte
value	Se referă la valorile CSS care se află după semnul două puncte

Majoritatea intrărilor sunt alcătuite dintr-un selector urmat de o pereche proprietate/valoare încadrată de acolade ({}). Selectorul definește domeniul de valabilitate al perechii proprietate/valoare care urmează, în timp ce perechea proprietate/valoare definește modul în care browserul va reda textul selectat cu ajutorul selectorului folosit. În câteva cazuri, descrierea prezintă și stiluri in-line.

Notă Numerele din parantezele care se află la sfârșitul fiecărui titlu de mai jos se referă la titlurile efective ale secțiunilor recomandărilor World Wide Web Consortium (W3C) pentru foi de stil, Cascading Style Sheets, Level 1 (CSS1), și Cascading Style Sheets, Level 2 (CSS2). Unele numere sunt sărite deoarece se referă la aspecte care nu au proprietăți.

Concepte de bază

Conținerea în HTML (CSS1 1.1)

```
<LINK rel="stylesheet" type="text/css" href="importedstylesheet " ... >
```

Foaie de stil externă.

```
@import importedstylesheet
```

Foaie de stil externă importată într-un element de stil in-line.

```
<STYLE type="text/css" ... ><- style sheet commands –></STYLE>
```

Foaie de stil in-line cu comenzi de stil conținute în comentariile SGML.

```
<STYLE><-- commands --></STYLE>
```

```
<XXX style=" ... ">
```

Comenzi de stil adăugate direct la un element HTML.

Gruparea (CSS1 1.2, CSS2 5.2.1)

```
element1, element2, element3 { property: value }
```

Economisiți spațiu în foile de stil prin gruparea elementelor care trebuie să fie tratate în același mod într-o listă separată prin virgule.

Moștenirea (CSS1 1.3)

```
element1 { property: value ...}
<element1> <element2> (same as outer) </element2> </element1>
```

Un element ia valorile elementului care îl conține, dacă nu este declarat în alt mod.

Class ca și selector (CSS1 1.4, CSS2 5.8.3)

```
element1.classid { property: value ... }
```



```
<element1 class="classid">
```

Elementele pot fi grupate și selectate după atributele de clasă declarate chiar în etichetele HTML sau XHTML. Dacă este omis numai numele elementului, atunci selectorul selectează toate elementele cu aceeași clasă. Notația punctată nu este disponibilă în documentele generale XML.

ID ca și selector (CSS1 1.5, CSS2 5.9)

```
#xxx { property: value ...}
element1#xxx { property: value ...}
<element1 id=#xxx>
```

Elementele pot fi adresate direct printr-un atribut unic ID în eticheta HTML individuală.

Selectori contextuali (CSS1 1.6)

```
element1 element2 { property: value ... }
<element1><element2>this text is affected</element2></element1>
<element2>this text is not</element2>
```

Domeniul de valabilitate al comenzilor poate fi limitat la elementele din domeniul de valabilitate al unei etichete care le conține.

Comentarii (CSS1 1.7)

```
/* comment */
```

Comentariile pot fi inserate în foaia de stil pentru a explica codul.

Selectori pentru descendenți (CSS2 5.5)[s2]

`<element1> * <element2>` Se potrivește cu un nepot sau cu un descendent mai îndepărtat.

Selectori pentru copii (CSS2 5.6)[s2]

`<element1> > <element2>` Se potrivește cu un copil (descendent direct).

Selectori pentru frați adiacenți (CSS2 5.7)[s2]

`<element1> + <element2>` Se potrivește cu un frate adiacent.

Potrivirea atributelor și a valorilor atributelor (CSS2 5.8.1)[s2]

`[att]`	Se potrivește atunci când elementul setează atributul „att", indiferent de valoarea atributului.
`[att=value]`	Se potrivește atunci când valoarea atributului „att" al elementului este exact „value".
`[att~=value]`	Se potrivește atunci când valoarea atributului „att" al elementului este o listă de „cuvinte" separate de spații, unul dintre ele fiind exact „value". Dacă se folosește acest selector, atunci cuvintele din „value" nu pot să conțină spații.
`[att|=val]`	Se potrivește atunci când valoarea atributului „att" al elementului este o

listă de „cuvinte" separate prin cratimă, începând cu „val". Căutarea începe de la începutul valorii atributului.

Pseudo-clase şi pseudo-elemente (CSS1 2, CSS2 5.10)

Pseudo-clasa first-child (CSS2 5.11.1)[s2]

E:first-child { *property*: *value* ... }[s2]

Se potrivește cu primul copil al oricărui element.

Pseudo-clasele anchor (CSS1 2.1, CSS2 5.11.2)

A:link { *property*: *value* ... }

A:active { *property*: *value* ... }[ns2]

A:visited { *property*: *value* ... }

Stilul ancorelor poate să depindă de setările browserului.

Pseudo-clasele dinamice (CSS2 5.11.2)[s2]

A:hover { *property*: *value* ... }[s2]

A:active { *property*: *value* ... }[s2]

A:focus { *property*: *value* ... }[s2]

Stilul ancorelor poate să depindă de setările dinamice ale browserului.

Pseudo-clasa de limbă (CSS2 5.11.4)[s2]

E:lang(xx) { *property*: *value* ... }[s2]

Stilul elementelor poate să depindă de limba care este utilizată.

```
HTML:lang(de) { quotes: '»' '«' '\2039' '\203A' }[s2]
:lang(de) > Q { quotes: '»' '«' '\2039' '\203A' }[s2]
```

Pseudo-elemente tipografice (CSS1 2.2, CSS2 5.12)

Pseudo-elementul :first-line (CSS2 5.12.1)[s2]

first-line (CSS1 2.3)[adv]

E:first-line { *property*: *value* ... }

Prima linie a oricărui element poate fi selectată direct.

Pseudo-elementul :first-letter (CSS2 5.12.2)[s2]

first-letter (CSS1 2.4)[adv]

E:first-letter { *property*: *value* ... }

Prima literă a oricărui element poate fi selectată direct.

Pseudo-elementele :before şi :after (CSS2 5.12.3)[s2]

E:before { content: *value* ... }[s2]

E:after { content: *value* ... }[s2]

Conținutul poate fi generat înainte sau după un element.

Pseudo-elemente în selectori (CSS1 2.5)

element1 element2:pseudo-element { *property*: *value* ... }

Pseudo-elementele trebuie să fie ultimele într-o listă de selectori contextuală.

Pseudo-elemente multiple (CSS1 2.6)

element1 { *property*: *value* ... }

element1:pseudo-element1 { *property*: *value* ... }

element1:pseudo-element2 { *property*: *value* ... }

Pot fi definite mai multe pseudo-elemente care au efecte care se suprapun.

Regula @import (CSS2 6.3)[s2]

@import url("pathname") modifier[s2]

Importă o foaie de stil externă. Comenzile @import trebuie să preceadă orice alte reguli ale foii de stil. Dacă în sursă există variante ale tipului media, modificatorul vă permite să le selectați astfel:

```
@import url("bigscreen.css") projection, tv;
```

Important (CSS1 3.1, CSS2 6.4.2)

element1 { ! *important property*: *value* ...}

Adaugă importanță unei comenzi de stil și o promovează deasupra comenzilor în conflict care, în caz contrar, ar înlătura-o. Există foarte puține situații în care folosirea acestei comenzi este justificată, fiind o metodă prin care o persoană cu necesități de vedere speciale poate să suprascrie foaia dumneavoastră de stil pentru o afișare alternativă cu un contrast mai mare sau cu caractere de mari dimensiuni. În trecerea la CSS2, W3C a efectuat o modificare importantă a ordinii în care se interpretează modificatorul, corectând astfel o greșeală de proiectare. În mod corect, CSS2 urmează o estimare a importanței făcută de utilizator înainte de părerea autorului. Până când suportul pentru CSS2 nu va fi universal, folosirea acestei comenzi de compensare nu este recomandată.

Ordinea cascadelor (CSS1 3.2, CSS2 6.4)

Sortarea importanței

Comenzile ! important sunt mai importante decât cele fără acest marcaj.

Sortarea originii

CSS1: Foile de stil ale autorului sunt mai importante decât foile de stil ale cititorului, care sunt mai importante decât foaia de stil prestabilită a browserului.

CSS2: Foile de stil ale autorului sunt mai importante decât foile de stil ale cititorului, care sunt mai importante decât foaia de stil prestabilită a browserului. Excepție face modificatorul ! important al cititorului, care este mai important decât unul al autorului. În acest fel, persoanele cu necesități speciale au un control mai bun asupra prezentării documentelor în browserul lor.

Sortarea specificității
: Rezolvă conflicte prin intermediul unui algoritm complex care încearcă să descopere cât de „specifică" este o comandă.

Sortarea ordinii
: Rezolvă conflictele rămase, prin intermediul unui algoritm ultimul-văzut. Dacă două comenzi sunt egale, ultima văzută are prioritate.

Regula @media (CSS2 7.2.1)[s2]

```
@media mediatype { stylesheet ruleset }
```

Delimitează setul de reguli al foii de stil după tipul de mediu suportat. Într-un singur document pot fi incluse mai multe foi de stil pentru a asigura suport unor medii diferite.

```
@import url("bigscreen.css") projection, tv;
```

Tipuri de medii recunoscute (CSS2 7.3)[s2]

Tabelul D.2 descrie tipurile de medii recunoscute în CSS2.

Tabelul D.2 Tipurile de medii CSS2

Tip mediu	Descriere
all	Potrivit pentru toate dispozitivele[s2]
aural	Destinat sintetizatoarelor vocale[s2]
braille	Destinat dispozitivelor Braille cu răspuns tactil[s2]
embossed	Destinat imprimantelor Braille[s2]
handheld	Destinat dispozitivelor hand-held[s2]
print	Destinat tipăririi în pagini din material opac și documentelor vizualizate în modul print preview[s2]
projection	Destinat prezentărilor proiectate pe un ecran[s2]
screen	Destinat în principal ecranelor color de calculator[s2]
tty	Destinat afișajelor care folosesc caractere de lățime fixă, cum sunt teleimprimatoarele, unele terminale de calculator mai vechi și dispozitivele portabile cu capacități de afișare limitate. Nu ar trebui să folosiți unități în pixeli cu tipul de mediu "tty".[s2]
tv	Destinat dispozitivelor de tip televizor[s2]

Grupuri de medii (CSS2 7.3.1)[s2]

Tabelul D.3 Grupurile de medii CSS2

Grup de medii	Tipurile de medii incluse în grup
continuous	aural, braille, handheld, screen, tty, tv[s2]
paged	embossed, handheld, print, projection, tv[s2]
visual	handheld, print, projection, screen, tty, tv[s2]
aural	aural, tv[s2]
tactile	braille, embossed[s2]
grid	braille, embossed, handheld, tty[s2]
bitmap	handheld, print, projection, screen, tv[s2]
interactive	aural, braille, embossed, handheld, screen, tty, tv[s2]
static	aural, braille, embossed, handheld, print, projection, screen, tty, tv[s2]

Folosite pentru definirea mediilor la care se aplică o anumită proprietate.

Proprietăți ale fontului (CSS1 5.2)

Proprietate: font-family (CSS1 5.2.2, CSS2 15.2.2)[visual]

```
element1 { font-family: value ...}
```

<family-name>
: Specifică un nume de familie de fonturi, cum ar fi Garamond sau Beppo.

<generic-family>

serif

sans-serif

cursive

fantasy

monospace

inherit[s2]
: Specifică o familie de fonturi prin caracteristici generale.

Proprietate: font-style (CSS1 5.2.3, CSS2 15.2.3)[visual]

normal

italic

oblique

inherit[s2]
: Specifică un tratament de text. Oblic este similar cu italic, dar nu face decât să încline caracterele. Italic veritabil modifică și forma caracterelor.

Proprietate: font-variant (CSS1 5.2.4, CSS2 15.2.3)[visual]

normal

small-caps

inherit[s2]
: Specifică un tratament de text. Majusculele reduse (small caps) sunt necesare pentru funcționalitatea de bază, dar se pot transforma în litere normale în browserele bazate pe text, iar browserele pot să ignore comanda pentru alfabete care nu sunt vest-europene, pentru care nu sunt definite majuscule sau pentru care tehnicile de subliniere a cuvintelor sunt diferite.

Proprietate: font-weight (CSS1 5.2.5, CSS2 15.2.3)[visual]

normal
bold
bolder
lighter
100-900
inherit[s2]

Specifică un tratament de text. Limbajul permite nouă grosimi, dar dacă sunt disponibile mai puține, acestea pot fi puse în corespondență cu cele care lipsesc după o logică anume.

Proprietate: font-stretch[visual][s2]

normal[s2]
wider[s2]
narrower[s2]
ultra-condensed[s2]
extra-condensed[s2]
condensed[s2]
semi-condensed[s2]
semi-expanded[s2]
expanded[s2]
extra-expanded[s2]
ultra-expanded[s2]
inherit[s2]

Specifică lățimea relativă a unui font.

Proprietate: font-size (CSS1 5.2.6)

<absolute-size>
xx-small
x-small
small
medium
large
x-large
xx-large
<relative-size>
larger
smaller
<length>
<percentage>
inherit[s2]

Specifică un tratament de text. Dimensiunea fontului poate fi specificată în valori absolute sau relative la dimensiunea curentă, după cum este mai ușor.

Proprietate: font-size adjust[s2]

<number>[s2]
none [s2]
inherit[s2]

Specifică dimensiunea la care trebuie să ajungă un font după ajustare. Numărul este raportul dintre dimensiunea fontului și x-height.

Proprietate: font (CSS1 5.2.7)

<font-family>
<font-style>
<font-variant>
<font-weight>
<font-size>
<font-height>
inherit[s2]

Specifică mai multe tratamente de text. Această proprietate este o notație prescurtată pentru `<font-style>`, `<font-variant>`, `<font-weight>`, `<font-size>`, `<font-height>` și `<font-family>`. Valorile pe care le poate lua fiecare pot fi introduse după această proprietate, fără să vă faceți griji care merge cu care. Browserul își va da seama. În CSS2 au fost adăugate cuvintele cheie *caption*, *icon*, *menu*, *message-box*, *small-caption* și *status-bar*, care se referă la grupurile de fonturi sistem.

Proprietăți de culoare și fundal (CSS1 5.3, CSS2 14.1)

Proprietate: COLOR (CSS1 5.3.1)[visual]

<color>
inherit[s2]

Specifică o culoare de prim-plan (text) pentru un element.

Proprietate: background-color (CSS1 5.3.2, CSS2 14.2.1)[visual]

<color>
transparent
inherit[s2]

Specifică o culoare de fundal pentru un element. Spre deosebire de HTML-ul tradițional, foile de stil vă permit să adăugați „spoturi de culoare" în spatele oricărui element vizibil. Rezultatul poate să fie foarte frumos sau foarte iritant, în funcție de cât de bine este folosită această capacitate.

Proprietate: background-image (CSS1 5.3.3, CSS2 14.2.1)[visual]

<url>
none
inherit

Specifică o imagine de fundal pentru un element.

Proprietate: background-repeat (CSS1 5.3.4, CSS2 14.2.1)[visual]

repeat
repeat-x
repeat-y
no-repeat
inherit[s2]

Specifică o valoare și o metodă de repetare. repeat acoperă aria disponibilă atât orizontal, cât și vertical. repeat-x repetă acoperirea numai pe orizontală. repeat-y repetă acoperirea numai pe verticală.

Proprietate: background-attachment (CSS1 5.3.5, CSS2 14.2.1)[visual]

scroll
fixed[adv]
inherit[s2]

Specifică modul în care se mișcă fundalul în timpul derulării. În mod prestabilit se mișcă împreună cu textul, dar puteți specifica, de exemplu, ca un logo să rămână centrat și imobil pe pagină în timp ce restul paginii defilează peste el.

Proprietate: background-position (CSS1 5.3.6, CSS2 14.2.1)[visual]

<percentage>
<length>
top
center
left
bottom
right
inherit[s2]

Specifică o poziție inițială pentru imaginea de fundal. Valorile pot fi combinate, iar dacă sunt prezente două valori numerice, valoarea pe orizontală se presupune că este prima. Deci background-position: 10pc 40pc ; va stabili imaginea la 10% față de marginea din stânga și 40% de sus.

Proprietate: background (CSS1 5.3.7)[visual]

<background-color>
<background-image>
<background-repeat>
<background-attachment>
<background-position>
inherit[s2]

Specifică simultan mai multe valori de fundal. Această proprietate este o notație prescurtată pentru <background-color>, <background-image>, <background-repeat>, <background-attachment> și <background-position>. Valorile pe care le poate lua fiecare pot fi introduse după această proprietate, fără să vă faceți griji care merge cu care. Browserul își va da seama.

Text[s2]

Proprietăți pentru text (CSS1 5.4, CSS2 16.4)

Proprietate: word-spacing (CSS1 5.4.1)[visual]

normal
<length>[adv]
inherit[s2]

Specifică spațiul prestabilit dintre cuvinte.

Proprietate: letter-spacing (CSS1 5.4.2)[visual]

normal
<length>[adv]
inherit[s2]

Specifică spațiul prestabilit dintre litere. Acest lucru este util pentru fonturile în stil vechi German Fraktur, în care emfaza este dată prin spațierea literelor, mai mare decât în mod normal.

Proprietate: text-decoration (CSS1 5.4.3, CSS2 16.3.1)[visual]

none
underline
overline
line-through
blink
inherit[s2]

Specifică un tratament de text. Textul *line-through* este ceea ce în mod normal se numește *strikeout* sau *strike-through* (text tăiat cu o linie). Acesta este util, printre altele, dacă doriți să dezactivați global sublinierea legăturilor.

Umbrirea textelor: proprietatea 'text-shadow' (CSS2 16.3.2)[visual][s2]

Proprietate: text-shadow[s2]

none[s2]
<color>? *<length>* *<length>* *<length>*?[s2]
inherit[s2]

Specifică unul sau mai multe tratamente de text prin umbrire. Lungimile sunt orizontală, verticală și una opțională de estompare. Umbrele succesive sunt separate de virgule. Nu contează dacă culoarea este înainte sau după lungimi.[s2]

Proprietate: vertical-align (CSS1 5.4.4, CSS2 10.8)[visual]

baseline
sub

super
top
text-top
middle
bottom
text-bottom
<length>[s2]
<percentage>
inherit[s2]

Specifică alinierea verticală a unui element față de textul înconjurător.

Proprietate: text-transform (CSS1 5.4.5, CSS2 16.5)[visual]

capitalize
uppercase
lowercase
none
inherit[s2]

Specifică un tratament de transformare a textului. Transformările sunt necesare pentru funcționalitatea de bază, din moment ce scrierea cu majuscule este un tratament de text de bază pentru majoritatea limbilor vest-europene, dar browserele pot să ignore comanda pentru alfabete care nu sunt vest-europene, pentru care nu sunt definite majuscule sau pentru care tehnicile de subliniere a cuvintelor diferă în mod substanțial.

Proprietate: text-align (CSS1 5.4.6, CSS2 16.2)[visual]

left
right
center
justify[adv]
string[s2]
inherit[s2]

Specifică un tratament de text. `justify` se referă la spațierea textului pentru a se potrivi precis atât față de marginea din stânga cât și față de marginea din dreapta, tip de aliniere care nu le place creatorilor de browsere deoarece este complicat și dificil.

Proprietate: TEXT-INDENT (CSS1 5.4.7, CSS2 16.1)[visual]

<length>
<percentage>
inherit[s2]

Specifică un tratament de text care apare la începutul unui element. Acesta este util, de exemplu, pentru indentarea primei linii dintr-un paragraf (sau pentru indentarea agățată). Paragrafele cu indentare agățată se întâlnesc des în materiale tehnice.

Proprietate: line-height (CSS1 5.4.8, CSS2 10.8)[visual]

normal
<number>
<length>
<percentage>
inherit[s2]

Specifică înălțimea liniei față de font. În termeni tipografici, aceasta se numește corpul textului. Deci ați putea stabili un font de 12 puncte pe un corp de 14 puncte pentru a obține o separare între linii. Sau ați putea seta același font pe un corp de 12 puncte pentru a înghesui rândurile. De obicei browserul va stabili textul cu un spațiu rezonabil, astfel încât să fie cât mai lizibil, dar puteți folosi această proprietate pentru a avea control complet, de exemplu pentru a lipi partea de jos a unei linii chiar de partea de sus a liniei următoare și a obține un efect tipografic special într-un logo sau într-un titlu.

Proprietăți pentru casete (CSS1 5.5)

Toate proprietățile următoare specifică modul în care sunt tratate casetele „invizibile" care înconjoară fiecare element, inclusiv vizualizarea lor cu ajutorul unor linii despărțitoare sau modificarea culorii de fundal. Folosirea spoturilor de culoare în fundal pentru a evidenția un titlu arată destul de prost dacă nu extindeți în același timp și caseta „invizibilă" în care este așezat titlul, pentru a permite fundalului să realizeze o bordură frumoasă în jurul său.

Marginile sunt partea de la limita exterioară a acestui mozaic imbricat. Acestea afectează spațiul alb din jurul unui element și sunt întotdeauna transparente pentru a permite să se vadă fundalul. Apoi vin bordurile. Acestea afectează spațiul din interiorul casetei și preiau caracteristicile fundalului setat pentru elementul însuși. Valorile pentru borduri vor determina dacă prin adăugarea unei borduri vizibile caseta „invizibilă" va deveni vizibilă. Aceasta, dacă este prezentă, apare între margine și chenarul casetei.

Proprietate: margin-top (CSS1 5.5.01, CSS2 18.3)[visual]

<length>
<percentage>
auto
inherit[s2]

Specifică marginea exterioară superioară a unui element.

Proprietate: margin-right (CSS1 5.5.02, CSS2 18.3)[visual]

<length>
<percentage>
auto
inherit[s2]

Specifică marginea exterioară dreapta a unui element.

Proprietate: margin-bottom (CSS1 5.5.03, CSS2 18.3)[visual]

<length>

<percentage>
auto
inherit[s2]
Specifică marginea exterioară inferioară a unui element.

Proprietate: margin-left (CSS1 5.5.04, CSS2 18.3)[visual]

<length>
<percentage>
auto
inherit[s2]
Specifică marginea exterioară stânga a unui element.

Proprietate: margin (CSS1 5.5.05, CSS2 18.3)[visual]

<length>
<percentage>
auto
inherit[s2]
Specifică una sau mai multe margini exterioare ale unui element. Dacă sunt specificate patru valori de lungime, se aplică în ordine: sus, dreapta, jos și stânga. Dacă există o singură valoare, aceasta se aplică pe toate laturile; dacă sunt două sau trei, valorile care lipsesc sunt luate de pe latura opusă.

Proprietate: padding-top (CSS1 5.5.06, CSS2 18.4)[visual]

<length>
<percentage>
auto
inherit[s2]
Specifică marginea superioară interioară a unui element. Aceasta stabilește spațiul dintre o bordură „invizibilă" a unui element și elementul însuși.

Proprietate: padding-right (CSS1 5.5.07, CSS2 18.4)[visual]

<length>
<percentage>
auto
inherit[s2]
Specifică marginea interioară dreapta a unui element. Aceasta stabilește spațiul dintre o bordură „invizibilă" a unui element și elementul însuși.

Proprietate: padding-bottom (CSS1 5.5.08, CSS2 18.4)[visual]

<length>
<percentage>
auto
inherit[s2]
Specifică marginea interioară inferioară a unui element. Aceasta stabilește spațiul dintre o bordură „invizibilă" a unui element și elementul însuși.

Proprietate: padding-left (CSS1 5.5.09, CSS2 18.4)[visual]

<length>
<percentage>
auto
inherit[s2]
Specifică marginea interioară stânga a unui element. Aceasta stabilește spațiul dintre o bordură „invizibilă" a unui element și elementul însuși.

Proprietate: padding (CSS1 5.5.10, CSS2 18.4)[visual]

<length>
<percentage>
auto
inherit[s2]
Specifică una sau mai multe margini interioare ale unui element. Acestea stabilesc spațiul dintre bordurile „invizibile" ale unui element și elementul însuși. Dacă sunt specificate patru valori de lungime, se aplică în ordine: sus, dreapta, jos și stânga. Dacă există o singură valoare, aceasta se aplică pe toate laturile, iar dacă sunt două sau trei, valorile care lipsesc sunt luate de pe latura opusă.

Proprietate: border-top-width (CSS1 5.5.11, CSS2 8.5.1)[visual]

thin
medium
thick
<length>
inherit[s2]
Specifică grosimea (și vizibilitatea) bordurii superioare a unui element.

Proprietate: border-right-width (CSS1 5.5.12, CSS2 8.5.1)[visual]

thin
medium
thick
<length>
inherit[s2]
Specifică grosimea (și vizibilitatea) bordurii din dreapta a unui element.

Proprietate: border-bottom-width (CSS1 5.5.13, CSS2 8.5.1)[visual]

thin
medium
thick
<length>
inherit[s2]
Specifică grosimea (și vizibilitatea) bordurii inferioare a unui element.

Proprietate: border-left-width (CSS1 5.5.14, CSS2 8.5.1)[visual]

thin
medium
thick
<length>
inherit[s2]

Specifică grosimea (și vizibilitatea) bordurii din stânga a unui element.

Proprietate: border-width (CSS1 5.5.15, CSS2 8.5.1)[visual]

thin
medium
thick
<length>
inherit[s2]

Specifică grosimea (și vizibilitatea) uneia sau mai multor borduri ale unui element. Această proprietate este o prescurtare pentru setarea 'border-top-width', 'border-right-width', 'border-bottom-width', 'border-left-width' dintr-o dată. Pot fi între una și patru valori, cu următoarele interpretări: o singură valoare – grosimea celor patru borduri va avea aceeași valoare; două valori – bordura de sus și cea de jos vor avea grosimea dată de prima valoare, iar bordura din dreapta și cea din stânga de a doua valoare; trei valori – sus prima valoare, dreapta și stânga a doua valoare, jos a treia valoare; patru valori – în ordine: sus, dreapta, jos și stânga.

Proprietate: border-color (CSS1 5.5.16, CSS2 8.5.2)[visual]

<color>
transparent[s2]
inherit[s2]

Specifică culoarea uneia sau mai multor borduri ale unui element. Pot fi între una și patru valori, cu următoarea interpretare: o singură valoare – culoarea celor patru borduri va avea aceeași valoare; două valori – bordura de sus și cea de jos vor avea culoarea dată de prima valoare, iar bordura din dreapta și cea din stânga de a doua valoare; trei valori – sus prima valoare, dreapta și stânga a doua valoare, jos a treia valoare; patru valori – în ordine: sus, dreapta, jos și stânga. Valoarea CSS2 `transparent` implică faptul că bordura este transparentă (invizibilă), deși ocupă în continuare loc pe pagină. Valoarea CSS2 `inherit` implică faptul că oricare ar fi valoarea setată într-un element strămoș, aceasta este explicit acceptată în acest descendent.

Proprietate: border-style (CSS1 5.5.17, CSS2 8.5.3)[visual]

none
dotted[adv]
dashed[adv]
solid
double[adv]
groove[adv]
ridge[adv]
inset[adv]
offset[adv]
inherit[s2]

Specifică stilul liniei uneia sau mai multor borduri ale unui element. Pot fi între una și patru valori, cu următoarea interpretare: o singură valoare – liniile celor patru borduri vor avea aceeași valoare; două valori – bordura de sus și cea de jos vor avea stilul de linie dat de prima valoare, iar bordura din dreapta și cea din stânga de a doua valoare; trei valori – sus prima valoare, dreapta și stânga a doua valoare, jos a treia valoare; patru valori – în ordine: sus, dreapta, jos și stânga.

Proprietate: border-top (CSS1 5.5.18, CSS2 8.5.4)[visual]

<border-top-width>
<border-style>
<color>
inherit[s2]

Stabilește simultan grosimea, culoarea și stilul de linie (numai) pentru bordura de sus.

Proprietate: border-right (CSS1 5.5.19, CSS2 8.5.4)[visual]

<border-right-width>
<border-style>
<color>
inherit[s2]

Stabilește simultan grosimea, culoarea și stilul de linie (numai) pentru bordura din dreapta.

Proprietate: border-bottom (CSS1 5.5.20, CSS2 8.5.4)[visual]

<border-bottom-width>
<border-style>
<color>
inherit[s2]

Stabilește simultan grosimea, culoarea și stilul de linie (numai) pentru bordura de jos.

Proprietate: border-left (CSS1 5.5.21, CSS2 8.5.4)[visual]

<border-left-width>
<border-style>
<color>
inherit[s2]

Stabilește simultan grosimea, culoarea și stilul de linie (numai) pentru bordura din stânga.

Proprietate: border (CSS1 5.5.22, CSS2 8.5.4)[visual]

<border-width>
<border-style>

<color>
inherit[s2]
Stabilește grosimea, culoarea și stilul de linie pentru toate bordurile în același timp și la aceleași valori. Nu puteți folosi valori multiple pentru a seta valori diferite pentru cele patru laturi.

Proprietate: width (CSS1 5.5.23, CSS2 10.2)[visual]

<length>
<percentage>
auto
inherit[s2]
Suprascrie lățimea prestabilită a unui element.

Proprietate: min-width (CSS2 10.4)[visual][s2]

<length>[s2]
<percentage>[s2]
inherit[s2]
Stabilește lățimea minimă a unui element.

Proprietate: max-width (CSS2 10.4)[visual][s2]

<length>[s2]
<percentage>[s2]
none[s2]
inherit[s2]
Stabilește lățimea maximă a unui element.

Proprietate: height (CSS1 5.5.24, CSS2 10.5)[visual][s2]

<length>
<percentage>[s2]
auto
inherit[s2]
Suprascrie înălțimea prestabilită a unui element.

Proprietate: min-height (CSS2 10.7)[visual][s2]

<length>[s2]
<percentage>[s2]
inherit[s2]
Stabilește înălțimea minimă a unui element.

Proprietate: max-height (CSS2 10.7)[visual][s2]

<length>[s2]
<percentage>[s2]
inherit[s2]
Stabilește înălțimea maximă a unui element.

Proprietate: float (CSS1 5.5.25, CSS2 9.5.1)[visual]

left
right
none
inherit[s2]
Suprascrie comportamentul prestabilit al unui element și face ca textul să curgă în jurul său ca și când ar fi o protuberanță pe marginea stângă sau dreaptă. De asemenea, suprascrie și proprietatea `display` și face ca elementul să se comporte ca un element bloc, indiferent de tip.

Proprietate: clear (CSS1 5.5.26, CSS2 9.5.2)[visual]

left
right
both
none
inherit[s2]
Stabilește comportamentul unui element față de elementele care plutesc (flotante), cu ajutorul unei liste care cuprinde laturile pe care elementul va refuza să plutească. `none` spune că elementul va pluti pe oricare latură. `both` spune că va trage întotdeauna spre spațiul liber cel mai apropiat de pe ambele margini. `left` spune că va trage spre marginea din stânga și va ignora dreapta, iar `right` viceversa.

Proprietăți de clasificare (CSS1 5.6)

Proprietate: display [adv] (CSS1 5.6.1, CSS2 5.6)[visual]

block
inline
list-item
none
run-in[s2]
compact[s2]
marker[s2]
table[s2]
inline-table[s2]
table-row-group[s2]
table-header-group[s2]
table-footer-group[s2]
table-row[s2]
table-column-group[s2]
table-column[s2]
table-cell[s2]
table-caption[s2]
inherit[s2]
Suprascrie comportamentul prestabilit al unui element, iar acesta se comportă cum doriți. Grozav. Din păcate, numai Netscape Navigator pentru

Windows 95 a folosit acestea până acum, și doar părțile din CSS1. S-ar putea ca Mozilla să fie cu mult mai bun.

Proprietate: white-space [adv] (CSS1 5.6.2, CSS2 16.6)[visual]

`normal`
`pre`
`nowrap`
`inherit[s2]`

Suprascrie comportamentul prestabilit al spațiilor albe ale unui element, astfel încât să se comporte așa cum doriți. Este o proprietate foarte utilă deoarece vă permite să tipăriți poezii sau alte texte în care este necesar un plasament exact al spațiilor albe, fără să pierdeți vremea cu spații fără separare și linii noi, dar păstrând sub control aspectul fontului. Din păcate, doar Netscape Navigator pentru Windows 95 și Mac folosește ceva din această proprietate.

Proprietate: list-style-type (CSS1 5.6.3, CSS2 12.6.2)[visual]

`disc`
`circle`
`square`
`decimal`
`decimal-leading-zero[s2]`
`lower-roman`
`upper-roman`
`lower-alpha`
`upper-alpha`
`lower-greek[s2]`
`lower-latin[s2]`
`upper-latin[s2]`
`hebrew[s2]`
`armenian[s2]`
`georgian[s2]`
`cjk-ideographic[s2]`
`hiragana[s2]`
`katakana[s2]`
`hiragana-iroha[s2]`
`katakana-iroha[s2]`
`none`
`inherit[s2]`

Stabilește tipul de enumerare prestabilit al unui element listă, similar cu modul în care atributul de tip poate fi folosit într-o etichetă de listă.

Proprietate: list-style-image (CSS1 5.6.4, CSS2 12.6.2)[visual]

<url>
`none`
`inherit[s2]`

Suprascrie tipul de enumerare al unui element listă, alegând o imagine care va fi folosită ca și marcaj de listă.

Proprietate: list-style-position (CSS1 5.6.5, CSS2 12.6.2)[visual]

`inside`
`outside`
`inherit[s2]`

Suprascrie modul de afișare prestabilit al unui element listă, specificând dacă numărul sau marcajul de listă va fi afișat prin indentare agățată, ca de obicei, sau in-line, ca și când ar fi o parte a textului.

Proprietate: list-style (CSS1 5.6.6, CSS2 12.6.2)[visual]

<keyword>
<position>
<url>
`inherit[s2]`

O notație prescurtată pentru stabilirea simultană a tuturor valorilor de stil ale listei.

Conținut generat, numerotare automată și liste (CSS2 12)[s2]

Proprietatea 'content' (CSS2 12.2)[all][s2]

Proprietate: content[s2]

<string>`[s2]`
<url>`[s2]`
<counter>`[s2]`
`attr(X) [s2]`
`open-quote[s2]`
`close-quote[s2]`
`no-open-quote[s2]`
`no-close-quote[s2]`
`inherit[s2]`

Se folosește împreună cu pseudo-elementele `:before` și `:after` pentru a genera conținut. `attr(X)` returnează valoarea atributului `X` al subiectului numai pentru selector. Contoarele pot fi generate fie prin funcția `counter()`, fie prin funcția `counters()`. În cazul în care conținutul depinde de mediu, se folosește o regulă `@media`.

Marcaje pentru citate (CSS2 12.4)[visual][s2]

Proprietate: quotes[s2]

<string><string> `...[s2]`
`none[s2]`
`inherit[s2]`

Se folosește pentru a stabili valoarea sau valorile marcajelor de început și

sfârșit ale unui citat precum și ale marcajelor imbricate. Aceste valori trebuie să fie folosite împreună cu o proprietate de conținut care specifică valori pentru deschiderea și închiderea citatului. Mai există două valori, `no-open-quote` și `no-close-quote`, care afectează nivelul de imbricare, dar nu au nici un efect vizibil, și care realizează eliziunea marcajelor de citat folosite în unele stiluri tipografice.

Contoare automate și numerotări (CSS2 12.5)[all][s2]

Proprietate: counter-reset[s2]

<identifier> *<integer>*? ...[s2]
none[s2]
inherit[s2]

Stabilește valoarea unuia sau mai multor contoare la zero sau la o valoare opțională *<integer>*.

Proprietate: counter-increment (CSS2 12.5)[all][s2]

<identifier> *<integer>*? ...[s2]
none[s2]
inherit[s2]

Stabilește valoarea cu care se incrementează unul sau mai multe contoare la unu sau la o valoare opțională *<integer>*.

Proprietate: counter(<identifier> 'list-style-type') (CSS2 12.5.2)[all][s2]

Proprietate: counters(<identifier> 'list-style-type'...) (CSS2 12.5.2)[all][s2]

Stabilesc tipul stilului listei pentru unul sau mai multe contoare.

Proprietate: marker-offset (CSS2 12.6.1)[visual][s2]

length[s2]
auto[s2]
inherit[s2]

Stabilește ofsetul marcajului unui contor al cărui element părinte are proprietatea `display` cu valoarea `marker`.

Medii paginate (CSS2 13)[s2]

Casete pentru pagini: regula @page (CSS2 13.2)[visual][paged][s2]

`@page` *<identifier>*? { pageproperty: pageattribute ; ... }

Definește o pagină.

Dimensiunea paginii: proprietatea 'size' (CSS2 13.2.2)[visual][paged][s2]

Proprietate: size[s2]

<length><length>?[s2]
auto[s2]
portrait[s2]
landscape[s2]
inherit[s2]

Stabilește dimensiunea și orientarea paginii.

Marcaje de decupare: proprietatea 'marks' (CSS2 13.2.3) [visual][paged][s2]

Proprietate: marks[s2]

crop[s2]
cross[s2]
none[s2]
inherit[s2]

Comută între marcajele de decupare sau aliniere.

Pseudo-clasele left, right, și first page (CSS2 13.2.4) [visual][paged][s2]

left pseudo-class[s2]

Exemplu: `@page` *<identifier>*`:left { property: value ... }`[s2]

Exemplu: `@page :left { property: value ... }`[s2]

Se potrivește cu un element de pe o pagină orientată spre dreapta într-un context cu pagini.

right pseudo-class[s2]

Exemplu: `@page :right { property: value ... }`[s2]

Se potrivește cu un element de pe o pagină orientată spre stânga într-un context cu pagini.

first pseudo-class[s2]

Exemplu: `@page :first { property: value ... }`[s2]

Se potrivește cu un element de pe prima pagină într-un context cu pagini.

Salturile la pagină nouă (CSS2 13.3)[visual][paged][s2]

Salt la pagină nouă înainte/după elemente: `'page-break-before'`, `'page-break-after'`, `'page-break-inside'` (CSS2 13.3.1)[visual][paged][s2]

Proprietate: page-break-before[s2]

auto[s2]
always[s2]
avoid[s2]
left[s2]
right[s2]
inherit[s2]

Stabilește comportamentul saltului la pagină nouă înainte de un element.

Proprietate: page-break-after[s2]

auto[s2]
always[s2]

`avoid[s2]`
`left[s2]`
`right[s2]`
`inherit[s2]`
Stabilește comportamentul saltului la pagină nouă după un element.

Proprietate: page-break-inside[s2]

`auto[s2]`
`avoid[s2]`
`inherit[s2]`
Stabilește comportamentul saltului la pagină nouă în interiorul unui element.

Utilizarea paginilor cu nume: 'page' (CSS2 13.3.2)[visual][paged][s2]

Proprietate: page

`<identifier>`
`auto`
Utilizează pagina cu nume.

Salturi la pagină nouă în interiorul elementelor: 'orphans', 'widows' (CSS2 13.3.3)[visual] [paged][s2]

Proprietate: orphans[s2]

`<integer>[s2]`
`inherit[s2]`
Stabilește controlul pentru orfan (primul rând al unui paragraf, care apare singur în partea de jos a unei pagini).

Proprietate: widows[s2]

`<integer>[s2]`
`inherit[s2]`
Stabilește controlul pentru văduv (ultimul rând al unui paragraf, care apare singur în partea de sus a unei pagini).

Tabele (CSS2 17)[s2]

Modelul tabelului (CSS 17.2) [s2]

Următoarele atribute cuvinte cheie CSS2 pot fi atribuite proprietății `display` descrise în secțiunile CSS1 5.6.1 și CSS2 9.2.5. Acestea sunt create pentru a vă permite să tratați orice element XML astfel încât să se comporte ca un tabel pentru afișare, dar sunt disponibile doar în CSS2.

`table[s2]` (În HTML: TABLE)
Specifică faptul că un element definește un tabel la nivel de bloc. Este un bloc dreptunghiular care participă într-un context de formatare de bloc.

`inline-table[s2]` (În HTML: TABLE)
Specifică faptul că un element definește un tabel la nivel in-line. Este un bloc dreptunghiular care participă într-un context de formatare in-line.

`table-row[s2]` (În HTML: TR)
Specifică faptul că un element este o linie de celule.

`table-row-group[s2]` (În HTML: TBODY)
Specifică faptul că un element grupează una sau mai multe linii.

`table-header-group[s2]` (În HTML: THEAD)
Asemănător cu `'table-row-group'`, dar, pentru formatare vizuală, grupul de linii este afișat întotdeauna înainte de orice altă linie sau grup de linii și după orice legendă de început. Agenții utilizatori care tipăresc pot să repete liniile de subsol pe fiecare pagină pe care se întinde tabelul.

`table-footer-group[s2]` (În HTML: TFOOT)
Asemănător cu `'table-row-group'`, dar, pentru formatare vizuală, grupul de linii este afișat întotdeauna după orice altă linie sau grup de linii și înainte de orice legendă de sfârșit. Agenții utilizatori care tipăresc pot să repete liniile de subsol pe fiecare pagină pe care se întinde tabelul.

`table-column[s2]` (În HTML: COL)
Specifică faptul că un element descrie o coloană de celule.

`table-column-group[s2]` (În HTML: COLGROUP)
Specifică faptul că un element grupează una sau mai multe coloane.

`table-cell[s2]` (În HTML: TD, TH)
Specifică faptul că un element este o celulă a unui tabel.

`table-caption[s2]` (În HTML: CAPTION)
Specifică legenda unui tabel.

Deci o foaie de stil CSS2 prestabilită pentru XHTML ar arăta cam așa:

```
TABLE     { display: table }
TR        { display: table-row }
THEAD     { display: table-header-group }
TBODY     { display: table-row-group }
TFOOT     { display: table-footer-group }
COL       { display: table-column }
COLGROUP  { display: table-column-group }
TD,TH     { display: table-cell }
CAPTION   { display: table-caption }
```

Obiecte anonime pentru tabele (CSS2 17.2.1)[s2]

Deoarece proprietățile de afișare ale tabelelor nu respectă întotdeauna modelul de afișare CSS2, obiectele anonime pentru tabele vor fi generate automat pentru a le conține dacă există elemente lipsă.

Selectori pentru coloane (CSS2 17.3)[s2]

Următoarele proprietăți se pot aplica pentru elemente coloană și grupuri de coloane:

Proprietate: border[s2]
Proprietate: background[s2]
Proprietate: width[s2]
Proprietate: visibility[s2]

Poziţia şi alinierea legendei (CSS2 17.4.1)[visual][s2]
Proprietate: caption-side[s2]

`top[s2]`
`bottom[s2]`
`left[s2]`
`right[s2]`
`inherit[s2]`
Stabileşte poziţia legendei.

Algoritmi pentru lăţimea tabelului: proprietatea 'table-layout' (CSS2 17.5.2)[visual][s2]
Proprietate: table-layout[s2]

`auto[s2]`
`fixed[s2]`
`inherit[s2]`
Stabileşte algoritmul folosit la organizarea tabelului.

Borduri (CSS2 17.6)[visual][s2]
Proprietate: border-collapse[s2]

`collapse[s2]`
`separate[s2]`
`inherit[s2]`
Stabileşte modelul folosit pentru borduri.

Modelul bordurilor separate (CSS2 17.6.1)[visual][s2]
Proprietate: border-spacing[s2]

`<length> <length>?[s2]`
`inherit[s2]`
Stabileşte distanţa dintre bordurile celulelor.

Proprietate: empty-cells[s2]

`show[s2]`
`hide[s2]`
`inherit[s2]`

Redarea audio a tabelelor (CSS2 17.7)[aural][s2]
Proprietate: speak-header[s2]

`once[s2]`
`always[s2]`
`inherit[s2]`
Stabileşte comportamentul redării auditive.

Interfaţa cu utilizatorul (CSS2 18)[s2]
Cursoare: proprietatea 'cursor' (CSS2 18.1)[visual][interactive][s2]
Proprietate: cursor[s2]

`<url>[s2]`
`auto[s2]`
`crosshair[s2]`
`default[s2]`
`pointer[s2]`
`move[s2]`
`e-resize[s2]`
`ne-resize[s2]`
`nw-resize[s2]`
`n-resize[s2]`
`se-resize[s2]`
`sw-resize[s2]`
`s-resize[s2]`
`w-resize[s2]`
`text[s2]`
`wait[s2]`
`help[s2]`
`inherit[s2]`
Stabileşte tipul cursorului. URL este o imagine. Cuvântul cheie este contextul în care se află cursorul.

Preferinţele utilizatorului pentru culori (CSS2 18.2)[visual][interactive][s2]

Orice proprietate de culoare poate primi următoarele atribute cuvinte cheie, în funcţie de culorile de sistem, ca valoare similară pentru o culoare cu nume. Aceasta vă permite să faceţi ca afişajul agentului utilizator să fie în culorile prestabilite ale interfeţei utilizatorului.[s2]

`ActiveBorder[s2]`
Bordura ferestrei active.

`ActiveCaption[s2]`
Legenda ferestrei active.

`AppWorkspace[s2]`
Culoarea de fundal a interfeţei pentru documente multiple.

`Background[s2]`
Fundalul suprafeței de lucru.

`ButtonFace[s2]`
Culoarea feței elementelor de afișare tridimensionale.

`ButtonHighlight[s2]`
Umbra închisă a elementelor de afișare tridimensionale (pentru marginile îndepărtate față de sursa de lumină).

`ButtonShadow[s2]`
Coloarea umbrei elementelor de afișare tridimensionale.

`ButtonText[s2]`
Textul de pe butoane.

`CaptionText[s2]`
Textul din legendă, caseta de redimensionare și săgețile din caseta de derulare.

`GrayText[s2]`
Textul gri (dezactivat). Această culoare este stabilită la `#000000` dacă driverul de afișare curent nu suportă o culoare gri uniformă.

`Highlight[s2]`
Element(e) selectat(e) într-un control.

`HighlightText[s2]`
Textul elementului(elor) selectat(e) într-un control.

`InactiveBorder[s2]`
Bordura ferestrei inactive.

`InactiveCaption[s2]`
Legenda ferestrei inactive.

`InactiveCaptionText[s2]`
Culoarea textului unei legende inactive.

`InfoBackground[s2]`
Culoarea de fundal pentru controalele tooltip.

`InfoText[s2]`
Culoarea textului pentru controalele tooltip.

`Menu[s2]`
Fundalul meniului.

`MenuText[s2]`
Textul din meniuri.

`Scrollbar[s2]`
Zona gri a barei de derulare.

`ThreeDDarkShadow[s2]`
Umbra închisă a elementelor de afișare tridimensionale.

`ThreeDFace[s2]`
Culoarea feței elementelor de afișare tridimensionale.

`ThreeDHighlight[s2]`
Culoarea de evidențiere pentru elementele de afișare tridimensionale.

`ThreeDLightShadow[s2]`
Culoarea deschisă pentru elementele de afișare tridimensionale (pentru marginile apropiate de sursa de lumină).

`ThreeDShadow[s2]`
Umbra închisă a elementelor de afișare tridimensionale.

`Window[s2]`
Fundalul ferestrei.

`WindowFrame[s2]`
Cadrul ferestrei.

`WindowText[s2]`
Textul din ferestre.

Contururi dinamice: proprietatea 'outline' (CSS2 18.4)[visual][interactive][s2]

Proprietate: outline[s2]

`<outline-color><outline-style><outline-width>``[s2]`
`inherit[s2]`
Proprietate prescurtată pentru stabilirea atributelor bordurilor. Contururile se aseamănă cu bordurile, dar nu ocupă spațiu în pagină.

Proprietate: outline-width[s2]

`<border-width>``[s2]`
`inherit[s2]`
Stabilește lățimea conturului. Vezi proprietățile bordurilor. Contururile se aseamănă cu bordurile, dar nu ocupă spațiu în pagină.

Proprietate: outline-style[s2]

`<border-style>``[s2]`
`inherit[s2]`
Stabilește stilul conturului. Vezi proprietățile bordurilor. Contururile se aseamănă cu bordurile, dar nu ocupă spațiu în pagină.

Proprietate: outline-color

`<color>``[s2]`
`invert[s2]`
`inherit[s2]`
Stabilește culoarea conturului. Vezi proprietățile bordurilor. Contururile se aseamănă cu bordurile, dar nu ocupă spațiu în pagină.

Foi de stil pentru proprietăți auditive (CSS2 19)[aural][s2]

Proprietăți de volum sonor: 'volume' (CSS2 19.2)[aural][s2]

Proprietate: volume[s2]

`<number>``[s2]`
`<percentage>``[s2]`

silent[s2]
x-soft[s2]
soft[s2]
medium[s2]
loud[s2]
x-loud[s2]
inherit[s2]

Stabileşte volumul sonor.

Proprietăţi de vorbire: 'speak' (CSS2 19.3)[aural][s2]

Proprietate: speak

normal[s2]
none[s2]
spell-out[s2]
inherit[s2]

Stabileşte stilul vorbirii.

Proprietăţi de pauză: 'pause-before', 'pause-after' şi 'pause' (CSS2 19.4)[aural][s2]

Proprietate: pause-before[s2]

<time>[s2]
<percentage>[s2]
inherit[s2]

Stabileşte valoarea pauzei dinainte de vorbire.

Proprietate: pause-after[s2]

<time>[s2]
<percentage>[s2]
inherit[s2]

Stabileşte valoarea pauzei de după vorbire.

Proprietate: pause[s2]

<time>[s2]
<percentage>[s2]
inherit[s2]

Proprietate prescurtată pentru stabilirea valorilor pause-before şi pause-after.

Proprietăţi de semnalizare: 'cue-before', 'cue-after' şi 'cue' (CSS2 19.5)[aural][s2]

Proprietate: cue-before[s2]

<url>[s2]
none[s2]
inherit[s2]

Stabileşte semnalul sonor dinainte de vorbire.

Proprietate: cue-after[s2]

<url>[s2]
none[s2]
inherit[s2]

Stabileşte semnalul sonor de după vorbire.

Proprietate: cue [s2]

<url>[s2]
none[s2]
inherit[s2]

Proprietate prescurtată pentru stabilirea semnalelor sonore cue-before şi cue-after.

Proprietăţi de mixare: 'play-during' (CSS2 19.6)[aural][s2]

Proprietate: play-during[s2]

<url>[s2]
mix?[s2]
repeat?[s2]
auto[s2]
none[s2]
inherit[s2]

Stabileşte volumul.

Proprietăţi spaţiale: 'azimuth' şi 'elevation' (CSS2 19.7)[aural][s2]

Proprietate: azimuth[s2]

<angle>[s2]
left-side[s2]
far-left[s2]
left[s2]
center-left[s2]
center[s2]
center-right[s2]
right[s2]
far-right[s2]
right-side[s2]
behind[s2]
leftwards[s2]
rightwards[s2]
inherit[s2]

Stabileşte poziţia orizontală simulată a sursei sonore sau a vorbitorului.

Proprietate: elevation[s2]

<angle>[s2]
below[s2]
level[s2]

above[s2]
higher[s2]
lower[s2]
inherit[s2]
Stabilește poziția verticală simulată a sursei sonore sau a vorbitorului.

Proprietăți caracteristice vocilor: 'speech-rate', 'voice-family', 'pitch', 'pitch-range', 'stress' și 'richness' (CSS2 19.8)[aural][s2]

Proprietate: speech-rate[s2]

<*number*>[s2]
x-slow[s2]
slow[s2]
medium[s2]
fast[s2]
x-fast[s2]
faster[s2]
slower[s2]
inherit[s2]
Stabilește ritmul vorbirii. Numărul este în cuvinte pe minut.

Proprietate: voice-family[s2]

<*specific voice*>[s2]
<*generic voice*>[s2]
inherit[s2]
Stabilește familia vocală. Vocile specifice sunt dependente de agentul utilizator.

Proprietate: pitch[s2]

<*frequency*>[s2]
x-low[s2]
low[s2]
medium[s2]
high[s2]
x-high[s2]
inherit[s2]
Stabilește înălțimea medie a vocii (frecvența). Bărbat mediu = 120 Hz, femeie = 210 Hz, copil = 300 Hz.

Proprietate: pitch-range[s2]

<*number*>[s2]
inherit[s2]
Stabilește intervalul de înălțimi ale vocii. Numărul este între 0 și 100. 50 = media.

Proprietate: stress[s2]

<*number*>[s2]
inherit[s2]
Stabilește un model de intonație. Numărul este între 0 și 100.

Proprietate: richness[s2]

<*number*>[s2]
inherit[s2]
Stabilește un model de sunet secundar. Numărul este între 0 și 100.

Proprietăți de rostire: 'speak-punctuation' și 'speak-numeral' (CSS2 19.9)[aural][s2]

Proprietate: speak-punctuation[s2]

code[s2]
none[s2]
inherit[s2]
Stabilește o punctuație rostită sau dedusă din pauze și intonație.

Proprietate: speak-numeral[s2]

digits[s2]
continuous[s2]
inherit[s2]
Stabilește rostirea numerelor cifră cu cifră (număr de telefon) sau naturală (sumă de bani).

Unități de măsură (CSS1 6)

Unitățile de măsură CSS1 și CSS2 urmează după numărul pe care îl caracterizează fără un spațiu intermediar. Deci 10px este corect, dar 10 px nu.

Unități de lungime (CSS1 6.1, CSS2 4.3.2)

em em-quad, dimensiunea în puncte a fontului curent.
ex *x*-height, înălțimea literei x din fontul curent.
px pixeli
in inci
cm centimetri
mm milimetri
pc procent
pt puncte, o unitate de măsură tipografică egală cu a șaptezeci și doua parte dintr-un inci. Cu alte cuvinte, într-un inci există 72 de puncte.

Unități procentuale (CSS1 6.2, CSS2 4.3.3)

<percentage>
Valorile procentuale nu sunt moștenite. Rezultatul unui calcul procentual este moștenit.

Unităţi pentru culoare (CSS1 6.3, CSS2 4.3.6)

`#000`
`#000000`
`(RRR,GGG,BBB)`
`(R%,G%,B%)`
`keyword`

Culorile din foaia de stil nu afectează imaginile.

URL-uri şi URN-uri (CSS1 6.4, CSS2 4.3.4)

`url("url")`

URL-urile sunt întotdeauna încadrate de paranteze şi pot fi înconjurate de apostrofuri sau ghilimele opţionale. Parantezele, virgulele, caracterele spaţiu, apostrofurile (') şi ghilimelele (") care apar într-un URL trebuie să fie introduse prin secvenţe escape împreună cu un backslash (\). URL-urile parţiale sunt interpretate faţă de sursa foii de stil, şi nu faţă de document. CSS2 introduce conceptul de URI, Uniform Resource Identifier, care este alcătuit din punct de vedere logic dintr-un URN, Uniform Resource Name, plus un URL. Totuşi, sunt în continuare accesate prin sintaxa `url("url")`.

Unghiuri (CSS2 4.3.7)[aural][s2]

`deg`	grade sexagesimale
`grad`	grade centezimale
`rad`	radiani

Unghiurile pot fi negative sau, dacă au un prefix „-“ sau „+“, relative.

Durate (CSS2 4.3.8)[aural][s2]

`ms`	milisecunde
`s`	secunde

Timpul, din păcate, nu poate fi negativ.

Frecvenţe (CSS2 4.3.9)[aural][s2]

`Hz`	Hertz
`kHz`	kiloHertz

Frecvenţele nu pot fi negative.

Şiruri (CSS2 4.3.10)[all][s2]

Şirurile pot fi scrise fie cu ghilimele, fie cu apostrofuri. Ghilimelele nu pot să apară în interiorul ghilimelelor decât ca secvenţe escape (ca `'\"'` sau ca `'\22'`). Şi apostrofurile se pot folosi în mod similar în interiorul apostrofurilor, ca `"\'"` sau `"\27"`.

Reprezentarea documentului CSS (CSS2 4.4)[all][s2]

`@charset "<ISO Identifier>"`

Referirea la caractere care nu sunt reprezentate într-o codificare de caractere (CSS2 4.4.1)[all][s2]

`\<hexvalue>`

Importanţa CSS

Foile de stil în cascadă reprezintă viitorul pe Web, deşi poate că vor fi în curând completate în XML cu XSL, Extensible Stylesheet Language (limbajul pentru foi de stil extensibile), sau cu alte iniţiative. Creatorii insistă să controleze toate variabilele dintr-o pagină, de la tipul de font până la spaţierea liniilor şi plasarea exactă a spaţiilor, exact la fel ca şi în cazul unei tipărituri. Nu au ajuns chiar în acest stadiu, dar se apropie de acesta cu fiecare versiune nouă de browser.

- Foile de stil sunt uşor de folosit şi sunt accesibile pentru toţi utilizatorii, pe când XSL este, într-o oarecare măsură, mai greu de înţeles pentru cei care nu sunt programatori, chiar dacă este mult mai puternic.
- Foile de stil sunt susţinute oficial de W3C, deci efectiv de toate organizaţiile membre ale W3C, printre care sunt şi majoritatea dezvoltatorilor mari de software din lume, deci în final vor fi suportate corect de toată lumea. Noi sperăm asta din tot sufletul.
- Foile de stil sunt disponibile atât pentru documente HTML cât şi XML, deci au o aplicabilitate mai largă dacă ne gândim la milioanele de pagini de date moştenite care există în prezent pe Web.
- Suportul pentru foi de stil este uşor de modularizat, astfel încât motoarele de redare pot încorpora acele porţiuni care se potrivesc pentru un anumit dispozitiv, deci este posibil să se folosească foi de stil şi pentru dispozitive hand-held sau telefoane mobile.

Toate aceste motive sunt destul de convingătoare, deci se pare că foile de stil vor fi suportate mai bine decât XSL, care la ora actuală este mai complicat şi mai puţin modularizat.

Teoria din spatele foilor de stil este simplă: puteţi crea o foaie de stil principală pentru întregul sit şi să suprascrieţi sau să extindeţi foaia de stil pentru oricare pagină, fără să modificaţi setările globale care nu vă interesează.

La fel, un utilizator poate să creeze o foaie de stil care să nu ţină cont de foaia de stil prestabilită pe care o foloseşte browserul când afişează paginile, permiţând în acelaşi timp proiectantului paginii să nu ţină cont de acele setări când va formata o anumită pagină.

Pentru proiectant, acest lucru înseamnă că va trebui să specifice dimensiunea fontului în termeni relativi şi să nu forţeze utilizatorul să accepte ideea sa de „destul de mare“. În general, sunteţi liber să vă jucaţi cu aproape orice şi puteţi realiza lucruri excelente cu ajutorul unor comenzi destul de simple.

Comenzi pentru foaia de stil

Comenzile din foaia de stil sunt simple şiruri de caractere, fiecare comandă fiind alcătuită dintr-un element HTML, un pseudo-element special sau o subclasă de elemente urmate de o listă de proprietăţi şi valori încadrată de acolade. Dacă sunt listate mai multe perechi proprietate/valoare, atunci trebuie să fie separate prin punct şi virgulă. Stilurile pot fi rafinate în continuare prin declararea contextului în care se aplică, prin listarea în ordine a elementelor sau prin aplicarea unor pseudo-elemente sau atribute care să modifice elementul HTML.

Deci stilul referitor la un element ancoră poate să fie modificat spunând că acel stil se aplică doar la elementele ancoră care sunt și legături, `A:link { color:blue }`, au fost vizitate, `A:visited { color:red }`, sau apar în domeniul unui paragraf, `P A { color:green }`.

Comenzile pot fi plasate într-un fișier separat, ceea ce este comod pentru controlul unui întreg sit dintr-un singur loc, în secțiunea de antet a fișierului, sau in-line, atașate la o etichetă individuală. În cazul unei comenzi de stil in-line, nu mai este necesară descrierea unui element deoarece stilul va afecta elementul la care este atașat. Analog, nu veți încadra stilul între acolade, ci doar veți pune comenzile într-o listă separată prin virgule, inclusă între semne de citat.

Această secțiune a tratat teoria luminoasă a CSS, dar acum să aruncăm o privire asupra realității sumbre a implementării teoriei în browserele actuale.

Suportul CSS în browsere

Ca să fim sinceri, suportul pentru CSS1 a fost până acum un eșec jalnic, iar CSS2 este doar o umbră a posibilităților sale. Cele câteva lucruri disponibile pe mai multe platforme aproape că nici nu merită să fie făcute din cauza dificultăților menținerii compatibilității pentru mai multe browsere, iar câteva combinații proprietate/valoare blochează browserele la intervale de timp aleatoare. Putem doar să sperăm că această situație se va îmbunătăți în timp. Avem motive să sperăm doar dacă creatorii de browsere vor trece în spatele standardelor existente și se vor concentra să implementeze corect *cel puțin* sintaxa definită și comenzile, înainte de a porni după „caracteristici" și „extensii" ciudate sau care nu fac parte din standard, pentru a atinge un fel de Nirvana mult sperată a diferențierii produselor.

Ați observat că am pus cei doi termeni, „caracteristici" și „extensii", des folosiți de cele două companii producătoare mai importante, între ghilimele. Opinia mea este că nu poți crea o extensie pentru un standard decât după ce ai reușit să îl implementezi corect, iar o extensie fără o bază stabilă și bine înțeleasă este doar un nume fantezist și necinstit pentru un bug.

În general, cel mai bun suport pentru mai multe platforme într-un browser important îl găsiți în Netscape Navigator/Communicator 4.0, deși versiunea beta de Mozilla, care va apărea ca Netscape 5.0, promite un suport mult îmbunătățit pentru CSS1 și CSS2. Acoperirea platformelor Windows și Macintosh este destul de echilibrată, iar ceea ce funcționează pe o platformă aproape sigur va funcționa și pe cealaltă. De asemenea, suportul pentru UNIX este destul de bun, deci sunt acoperite toate platformele mai importante. Deoarece mai au încă un segment important din piață, iar utilizatorii Macintosh se știe că se pierd repede în fața unor teme ca designul paginilor sau preferințelor tipografice, aceasta este platforma cea mai potrivită pentru realizarea unui numitor comun, în ciuda penetrării pe piață de 55% a MSIE, în care intră atât versiunile noi cât și cele vechi ale browserului Microsoft.

Pe de altă parte, dacă sunteți interesat numai de Windows 95, Internet Explorer 4.0 de la Microsoft acoperă bine Windows și Intel PC. Din nefericire, implementarea lor CSS1 pentru Macintosh este groaznică, iar suportul lor UNIX este în cel mai bun caz o idee venită prea târziu, iar în cel mai rău un afront.

Dacă sunteți dispus să mergeți împotriva curentului principal, versiunea GNU a lui

Emacs-W3 (GnuScape Navigator) este probabil cea mai completă și mai actuală implementare disponibilă a CSS1. Deși configurarea Gnu Emacs-W3 nu este pentru cei slabi de înger, deoarece presupune să aveți buzunarele pline cu stilouri și creioane colorate, este de departe cel mai puternic și extensibil browser Web existent. Are încorporate numeroase caracteristici, inclusiv o integrare rapidă cu Emacspeak, un excelent pachet de browser auditiv de T.V. Raman, foarte potrivit pentru navigarea pe Web fără ajutorul văzului.

Deoarece Emacs-W3 este creat în jurul unui editor, importul informațiilor de pe Web și încorporarea lor în documente devine o sarcină ușoară. Deoarece este Emacs, puteți realiza filtrări și aranjări sofisticate ale datelor de pe Web, fără să vă luptați la fiecare pas cu ceea ce au crezut creatorii că *trebuie* să faceți.

Gnu Emacs-W3 poate fi găsit la `http://www.cs.indiana.edu/elisp/w3/docs.html`.

Programul Emacspeak al lui T.V. Raman poate fi găsit la `http://cs.cornell.edu/home/raman/emacspeak/emacspeak.html`.

Pe Web lucrurile se mută uneori, deci o vizită rapidă la un motor de căutare vă poate duce adesea la sursa actuală atunci când un document a „dispărut". Majoritatea paginilor utile se reîncarnează rapid după ce au dispărut de la locația anterioară.

Acum, după ce am văzut problemele asociate cu CSS în lumea reală, să ne gândim cum să rezolvăm problemele inevitabile, sau dacă într-adevăr dorim să le rezolvăm.

Rezolvarea elegantă a bug-urilor

Nu sunt foarte hotărâtă cum să abordez acest subiect. Pe de o parte, nu este corect ca proiectanții de pagini Web să petreacă nenumărate ore încercând să țină cont de capriciile și neglijențele producătorilor.

Mă gândesc adesea că ar trebui să codăm pentru modul în care *ar trebui* să funcționeze și să lăsăm oamenii să vadă cât de proaste sunt majoritatea browserelor. Ce-o fi o fi.

Pe de altă parte, oamenii vor judeca pagina dumneavoastră în funcție de cât de bine ați folosit instrumentele pe care le-ați avut la dispoziție, iar folosirea unor caracteristici care blochează unele browsere face pagina să arate prost. Puțini sunt destul de experimentați pentru a-și da seama că pagina groaznică pe care încearcă să o citească se datorează incapacității producătorului browserului cu care au deschis pagina, și nu erorilor creatorului paginii. Așa cum celor care au nevoie de ochelari noi tipăriturile li se par din ce în ce mai neclare, celor care au nevoie de browsere mai bune li se pare că paginile devin din ce în ce mai confuze.

În practică, asta înseamnă că va trebui mai întâi să realizați cea mai bună treabă posibilă cu etichetele tradiționale și să aplicați o foaie de stil abia după aceea. Acest lucru are sens, deoarece există încă numeroase browsere pentru care foaia dumneavoastră de stil pare o bolboroseală fără sens și trebuie să țineți cont și de acestea.

Asta înseamnă că va trebui să argumentați elementele tipografice de mai multe ori, apelând la o culoare de fundal, o imagine și culori pentru legături în eticheta `<BODY>`, ca și în selectorul `body`, și așa mai departe.

Înseamnă, de asemenea, că veți apela la lucruri care nu afectează drastic aspectul paginii. Cererea unui anumit font, de exemplu, este un amănunt de finețe, dar eșecul *folosirii* efective a acelui font nu va afecta lizibilitatea textului, decât dacă este cumva în chirilică sau ceva asemănător.

Referinţe

Deşi informaţiile din această anexă sunt actualizate la momentul scrierii, sursa cu cele mai proaspete şi corecte informaţii este World Wide Web Consortium, la `http://www.w3.org/Style/`.

Acest director conţine atât ultimele concepţii legate de foile de stil în cascadă, cât şi proiecte de cercetare care vă arată tendinţele viitoare.

În plus, capacităţile browserelor se modifică rapid, iar ambii producători importanţi sunt hotărâţi să suporte foile de stil. Deci vă puteţi aştepta ca o parte din caracteristicile pe care le-am marcat ca fiind nesigure în utilizare să fie mai puţin nesigure în viitor. Recomandarea mea personală este să luaţi un marker galben şi să subliniaţi elementele pe care le-aţi testat personal şi s-au dovedit sigure sau care v-au fost recomandate ca sigure de o anumită sursă credibilă. În majoritatea cazurilor, eşecul satisfacerii unei comenzi din foaia de stil nu va avea alt efect decât să deranjeze puţin formatarea. Însă în cazurile cele mai grave, formatarea va fi complet distrusă iar documentul va fi redat într-o formă ilizibilă. Este posibil chiar să blocheze browserul.

Sugestie de la LeeAnne.com Words to weave by

Amintiţi-vă întotdeauna să folosiţi excelentul serviciu de validare CSS2 al W3C, disponibil la `http://jigsaw.w3.org/css-validator/`, înainte de utilizarea (sau chiar testarea) unei foi de stil. Puteţi folosi de asemenea serviciul lor de validare HTML/XHTML la `http://validator.w3.org/` înainte de a pierde prea mult timp cu testarea. Întotdeauna este enervant când sursa unei probleme este o simplă eroare de codare, iar folosirea instrumentelor de validare este o foarte bună practică de proiectare şi codare.

Pe scurt, foile de stil nu sunt chiar gata de consacrare, decât în cazul în care aveţi destul timp liber pentru a regla şi aranja paginile la fiecare apariţie a unei noi implementări, care ar trebui să aibă mai puţine bug-uri.

Note asupra CSS2, generaţia următoare a foilor de stil

CSS2, versiunea următoare a foilor de stil, preia câteva caracteristici ale specificaţiilor existente şi le extinde prin includerea mai multor pseudo-elemente, care permit un control mult mai bun al documentului, şi prin adăugarea unor proprietăţi specifice care influenţează redarea în browserele auditive.

Aceste modificări au apărut ca urmare a importanţei din ce în ce mai mari a obţinerii de date din reţea în viaţa de zi cu zi. Pe măsură ce browserele Web devin omniprezente, apărând peste tot, de la maşinile de lux la sistemele video de acasă, posibilităţile de redare optimă a informaţiei se înmulţesc dincolo de trucurile simple pe care le realizăm acum.

Dacă sunteţi sub o maşină şi încercaţi să reparaţi o defecţiune după instrucţiunile de pe situl Web al fabricantului, probabil că veţi dori să auziţi textul, iar imaginile să fie cât mai mari ca să le puteţi vedea uitându-vă pe după o roată şi cu gâtul într-un unghi incomod.

În maşina care circulă cu viteză mare pe şosea, nu doriţi ca atenţia să vă fie distrasă. Poate chiar maşina va aplica o foaie de stil care restrânge ieşirea la materialul vorbit, atunci când vă aflaţi în mişcare, dar vă lasă să consultaţi hărţi şi descrieri de tip text atunci când este complet oprită.

Pentru un chirurg care caută fişe medicale în mijlocul unei operaţii, o redare auditivă telegrafică şi rapidă poate fi necesară pentru a fi la curent cu exigenţele medicale ale momentului.

Putem doar să ne imaginăm cum va arăta viitorul peste câţiva ani, dar cred că este destul de sigur că Web-ul va fi prezent în continuare sub o formă sau alta şi că va fi integrat în viaţa noastră zilnică în moduri pe care ni le putem doar vag imagina astăzi.

Termeni folosiţi în această carte

Utilizarea acestui glosar

Când veţi citi definiţiile din glosarul de mai jos, veţi întâlni cuvinte sau expresii scrise cu **aldine**. Acestea se referă la intrări în glosar care pot conţine informaţii înrudite sau suplimentare. Elementul referit poate să clarifice legătura definiţiei cu alte definiţii din glosar sau să extindă înţelesul intrării trimiţându-vă la o definiţie mai generală sau mai particulară. Poate, de asemenea, să vă indice un sinonim sau un antonim.

Glosar

accesibil Care poate fi vizualizat de persoane într-o mare varietate de circumstanţe. Deşi cuvântul are un înţeles legal şi cerinţe specifice formulate în Americans with Disabilities Act (Legea pentru americanii cu handicap), se aplică de fapt tuturor. De exemplu, informaţiile trebuie să fie la fel de accesibile pentru utilizatorii Mac-urilor, PC-urilor sau ai sistemelor Unix, sau celor care folosesc un monitor monocolor pentru arte grafice. *Vezi şi* **design accesibil**.

analizor Un program care înţelege sintaxa unui limbaj şi poate citi un fişier de intrare, furnizând la ieşire producţii intermediare sau fluxuri de date modificate, pe care un alt program le poate utiliza. Unele analizoare XML pot de asemenea să **valideze** fişierul de intrare în funcţie de o **schemă** de definiţie.

anticipare Într-un analizor, capacitatea de a citi înainte şi de a reveni dacă găseşte o ambiguitate pe care nu o poate rezolva euristic examinând fluxul de intrare într-o manieră care nu este secvenţială. XML nu este capabil de anticipare.

apariţii În **XML Schema**, numărul de apariţii ale unui element sau model de conţinut.

aplicaţie Nivelul software care primeşte ieşirea de la un procesor XML şi face ceva util cu aceasta.

arbore al documentului Modelul abstract al unui document ierarhic care permite ca nodurile individuale, elementele şi secţiunile să fie modificate prin instrucţiuni de prelucrare şi de stil.

arbore de obiecte flux Mulţimea de obiecte flux **XSL** care alcătuiesc un **document XML**.

arc În **RDF**, reprezentarea unei muchii **predicat** sau **proprietate** în forma de arbore orientat.

atribut Informaţie conţinută în interiorul unei etichete ca pereche `name="value"` şi nu ca un conţinut între eticheta de pornire şi cea de încheiere. O **etichetă de element vid** nu conţine alte informaţii decât cele din atributele sale.

ATTLIST Într-un DTD, marcaj care defineşte o **listă de atribute**. `<!ATTLIST attribute datatype {"default"} >`

blocuri înlocuitoare Două blocuri de secvenţe de caractere pe 16 biţi rezervate pentru codificarea unor caractere obscure sau rar utilizate. Acestea sunt în intervalele `[#xD800-#xDBFF]` şi `[#xDC00-#xDFFF]` şi sunt utilizate în perechi, câte unul din fiecare bloc.

cadru În **CORBA**, modulul de comunicaţie care realizează interfaţa dintre nucleul **ORB** şi implementarea **serverului**.

caracter Orice caracter Unicode / ISO 8879 / ISO 10646, în afară de caracterele din **blocuri înlocuitoare**, `#xFFFE` şi `#xFFFF`. Caracterele permise sunt `#x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]`. Pentru un ghid excelent şi ieftin în lumea minunată a caracterelor permise, vă recomand cu căldură să achiziţionaţi Unicode Standard Version 3.0, o carte scrisă competent, bună de pus pe măsuţa de cafea, cu CD-ROM inclus, disponibilă la `http://www. unicode.org/`, şi care va face plăcere celor interesaţi de limbi străine şi sisteme de scriere. De asemenea, este un ghid de referinţă pentru simbolurile matematice şi ştiinţifice utilizate în mod obişnuit în lumea întreagă.

care validează *Vezi* **validare**.

Cascading Style Sheets (CSS – foi de stil în cascadă) O metodă de separare a conţinutului de prezentare bazată pe contextul etichetei şi poziţia sa în **arborele documentului**. CSS foloseşte un model ierarhic al documentului pentru a influenţa prin moştenire modul de **redare** a conţinutului paginii într-un browser.

client Un obiect distribuit care trimite cereri unui **server** şi primeşte răspunsuri.

comentariu Într-un **DTD** sau într-un document **XML**, text explicativ care nu trebuie analizat şi care nu implică nici o acţiune. `<!-- comment text -->`

compunere În **XML Schema**, metodă utilizată pentru a construi o schemă completă din componente ale mai multor zone de nume.

constrângere În **XML Schema**, limitele impuse pentru valorile pe care le poate lua un element sau atribut. În **XML** în general, orice limitare impusă unui conţinut, inclusiv **construirea corectă** şi **validitatea**.

construit corect Un **document XML** imbricat corect, cu o structură de etichetare corectă şi care satisface şi alte criterii definite în Recomandarea **XML** 1.0 de la **W3C**. Este posibilă generarea unui **DTD** pornind de la orice document construit corect, care prin urmare va deveni valid, deşi nu orice document construit corect este valid ca o instanţiere a unui anumit DTD. *Vezi şi* **valid**.

context În **CSS** şi **XML Schema**, lista secvenţială a elementelor părinte care localizează orice nod din arborele documentului.

control al exportului În **XML Schema**, orice restricţii impuse asupra disponibilităţii unei constrângeri pentru un proces exterior.

conţinut Fie un **tip de date**, fie un **model de conţinut**.

conţinut ANY Într-un DTD XML, un cuvânt cheie pentru declararea elementelor, care indică faptul că un element poate să conţină orice tip de marcaje şi date, în orice combinaţie. `<!ELEMENT elementname ANY>`

conţinut element Într-un DTD, un model de conţinut care permite doar alte elemente în interiorul unui element.

conţinut mixt Într-un DTD, un model de conţinut care permite ca un element să

conține atât date caracter, cât și alte elemente.

conținut vid *Vezi* **element vid.**

copil În **XPath** și în **XML** în general, descendentul imediat al unui nod. *Vezi și* **descendent.**

CORBA (Common Object Request Broker Architecture – arhitectura brokerului de cerere de obiecte comune) Un standard orientat spre obiecte, permițând aplicațiilor distribuite să comunice între ele.

crâng (grove) O vedere canonică asupra arborelui sau grafului orientat al informațiilor, în care fiecare nod este o proprietate a mulțimii de informații și există un singur nod rădăcină. Conceptul este foarte des folosit în **DSSSL, SGML** și **HyTime** și corespunde aproximativ unui DOM în **XSL**, dar nu are un echivalent exact. Un crâng este în esență o vedere asupra informației obiect care poate să difere prin tipul notației folosite pentru a-l descrie și poate să servească pentru a lega diferite DTD-uri XML (zone de nume) care pot să descrie aceleași mulțimi de date. Mai este cunoscut și sub numele de **pădure.**

date caracter Textul unui document XML, în afara marcajelor; ceea ce rămâne după eliminarea din document a tuturor etichetelor, instrucțiunilor de prelucrare și declarațiilor XML. Cu alte cuvinte, textul brut, înainte de aplicarea oricărui marcaj.

declarație În **XML Schema**, o instrucțiune prin care se creează în document un element sau un atribut cu un nume specific și un anumit tip.

declarație de autonomie O înștiințare teoretică pentru o aplicație că nu există declarații de marcare externe care să influențeze datele transferate de la procesorul XML la aplicație. Indiferent de situație, nu există nici o acțiune obligatorie. `<?xml standalone= "no" | "yes"?>`

declarație de codificare Într-un DTD sau într-o entitate externă, o valoare ISO care identifică o codificare a caracterelor.

declarație de entitate Într-un DTD, un nume urmat fie de un șir de înlocuire, fie de un ID public și un URL care indică spre declarații XML externe. `<ENTITY name "pubid" "url">`

declarație de tip de document *Vezi* **DOCTYPE.**

declarație DTD În general în **XML**, instrucțiunea care conține submulțimea internă a DTD-ului și/sau indică submulțimea externă a DTD-ului.

declarație XML Instrucțiunea opțională de la începutul unui fișier XML care identifică fișierul ca document XML de o anumită versiune, în momentul de față aceasta fiind întotdeauna „1.0", putând identifica de asemenea și setul de caractere folosit pentru codificare și dacă documentul depinde de fișiere externe. `<?xml version="1.0" … ?>`

definirea tipurilor documentului *Vezi* **DTD.**

definiție În **XML Schema**, o instrucțiune prin care se creează un nou **prototip** sau un nou **tip de date.**

dereferențiere Înlocuirea unui pointer cu valoarea la care se referă pointerul. În **XML Schema**, citirea și prelucrarea unui fișier schemă extern. Deși în programare este un termen perfect legitim, în domeniul larg al standardelor XML este mai mult un element de jargon. *Vezi și* **includere.**

descendent În **XPath** și în general, oricare nod din secvența continuă de noduri **copil** care se ramifică dintr-un **nod.** Altfel spus, toate elementele conținute într-un element dat.

design accesibil În mare, un design fără egoism, care nu presupune că proiectantul este centrul universului și că întruchipează în acest mediu suma hardware-ului, software-ului, abilităților și deprinderilor pe care toată lumea ar trebui să le aibă sau să aspire să le aibă. În sens restrâns, design de prezentare care permite persoanelor cu handicap motor, de vedere sau auditiv să folosească un document fără bariere artificiale. *Vezi și* **accesibil.**

DOCTYPE Într-un document **XML**, o instrucțiune care declară numele documentului și identifică opțional DTD-ul care definește formatul și conținutul documentului. `<!DOCTYPE name "public id" "url" [ XML declarations ]>`

document Orice entitate distinctă care conține informații, **documentul XML** fiind doar un exemplu.

document XML Un document care descrie un arbore abstract de informații, creat în conformitate cu regulile din Recomandarea XML de la W3C. Documentele XML extind și generalizează conceptul familiar de etichetare, așa cum este acesta folosit în HTML 4.0, pentru a include informații de context și de sens complete. Documentele XML pot fi **construite corect**, ceea ce înseamnă că sunt corecte din punct de vedere sintactic, sau pot fi **valide**, ceea ce înseamnă că respectă regulile unui **DTD** existent.

domeniu de valabilitate În **XML Schema** și în alte contexte, intervalul de validitate a numelui unui element. De asemenea, instrumentul **XML Namespace** care permite elementelor să se amestece fără conflicte de valabilitate. *Vezi și* **zone de nume.**

DSSSL (Document Style Semantics and Specification Language – limbajul de specificare și semantică de stil a documentului) Un limbaj pentru foi de stil, foarte puternic și complex, creat inițial pentru **SGML.**

DSSSL-o O versiune redusă a DSSSL, creată pentru o utilizare ușoară on-line împreună cu XML.

DTD (Document Type Definition – definirea tipurilor documentului) Colecția de declarații **XML** care definește formatul și conținutul unui document **XML.**

Dublin Core Un cadru de descriere a resurselor dezvoltat în Dublin, Ohio. Face parte din multele serii de **triplete** definite și poate fi folosit pentru o **instrucțiune** despre un obiect **resursă** în **RDF.** Utilizat și ca **metadate** în documentele HTML sau **XHTML** existente.

el însuși În **XPath, nodul** în sine. Nodul context din care sunt selectate celelalte noduri.

ELEMENT În XML, o etichetă sau numele din declarația sa de definiție dintr-un DTD.

element document *Vezi* **rădăcină.**

element vid Un element fără conținut in-line.

entitate externă O entitate, analizată sau neanalizată, generală sau parametru, localizată în afara **documentului XML** în sine.

entitate generală O **entitate** de forma `&name;` care poate fi folosită în interiorul conținutului unui **document XML.** *Vezi și* **entitate parametru.**

entitate internă O **entitate** definită în **submulțimea internă a unui DTD** sau una dintre cele predefinite în **XML.**

entitate neanalizată O entitate binară sau o altă entitate pe care un procesor XML nu o înțelege. Exemplele cele mai obișnuite sunt imaginile, secvențele audio și video și alte fișiere multimedia, fișiere în limbaje de programare și fișiere obiect executabile.

entitate parametru Într-un **DTD**, o entitate de forma `%name;` folosită numai în interiorul DTD-ului. Termenul și forma nu au nici un sens în afara DTD-ului. *Vezi și* **entitate generală.**

entitate predefinită Câteva entități caracter care sunt atât de importante în prelucrarea XML încât entitățile generale care le corespund sunt predefinite. Acestea sunt "<", ">", "&", "'" și '"'. Teoretic, ele pot fi folosite fără a le declara astfel: `&`, `<`, `>`, `'` și `"`. Pentru compatibilitate cu procesoare mai vechi, ar trebui oricum să le declarați. Declarațiile arată astfel: `<!ENTITY lt "<">`, `<!ENTITY gt ">">`, `<!ENTITY amp "&">`, `<!ENTITY apos "'">` și `<!ENTITY quot """>`.

entitate text În **XML**, o entitate de forma `&name;` care poate fi utilizată ca un înlocuitor mnemonic pentru un caracter sau un șir.

ENTITY Orice unitate de stocare care alcătuiește orice parte a unui **document XML.**

eroare O încălcare a regulilor unei specificații, care poate fi detectată și raportată aplicației și din care se poate reveni și se poate continua prelucrarea. *Vezi și* **eroare fatală.**

eroare fatală O eroare pe care un procesor XML trebuie să o detecteze și să o raporteze aplicației și din care nu se mai poate reveni. *Vezi și* **eroare.**

etichetă de element vid Marcarea unui element vid printr-o singură etichetă alcătuită dintr-o combinație de **etichetă de pornire** și **etichetă de încheiere** care adaugă un spațiu și un caracter slash final la element, astfel: `<elementname />`

etichetă de încheiere Eticheta care încheie un element, și care conține numele elementului precedat de un caracter slash și încadrat de paranteze unghiulare: `</elementname>`. Există o formă specială de etichetă de încheiere în cazul **elementelor vide,** alcătuită dintr-o combinație de etichetă de pornire și etichetă de încheiere care adaugă un spațiu și un caracter slash final la element, astfel: `<elementname />`. *Vezi și* **etichetă de element vid.**

etichetă de pornire Eticheta care deschide un element și care conține, opțional, setări de atribute: `<elementname {attributes} >`. Există o formă specială de etichetă de pornire în cazul **elementelor vide,** alcătuită dintr-o combinație de etichetă de pornire și etichetă de încheiere care adaugă un spațiu și un caracter slash final la element, astfel: `<elementname />`. *Vezi și* **etichetă de element vid.**

export În **XML Schema,** a face disponibilă o constrângere pentru un proces exterior.

Extensible Stylesheet Language *Vezi* **XSL.**

fațetă În **XML Schema,** o fațetă este o caracteristică definitorie pentru un concept sau un obiect.

foaie de stil O listă de instrucțiuni care asociază elementelor documentului proprietăți de afișare.

fragment O porțiune dintr-un document XML. Fragmentele de documente trebuie să fie construite corect.

fragment de document Orice porțiune a unui document **XML.**

frate În **XPath,** un alt **nod** care are același **părinte.**

GIOP (General Inter-ORB Protocol – protocolul general de comunicare între brokerii de cerere de obiecte) În **CORBA,** protocolul independent de mediu pentru comunicații între **ORB-uri.** *Vezi* **IIOP.**

grup de atribute În **XML Schema,** mai multe **atribute** care pot fi tratate ca o unitate refolosibilă de declarații de atribute.

grup de modele În **XML Schema,** mai multe modele de conținut care pot fi tratate ca o unitate refolosibilă.

hiperlegătură Mecanismul de legare fundamental în World Wide Web. O referință la o memorie îndepărtată.

HTML (Hypertext Markup Language – limbajul de marcare hipertext) Este o aplicație a **SGML** proiectată pentru a **reda** și a **hiperlega** documente pe World Wide Web.

HyTime (Hypermedia Time-based Structuring Language – limbajul de structurare hipermedia bazat pe timp) ISO/IEC 10744 este o aplicație sau o îmbunătățire a SGML care extinde posibilitățile SGML pentru sincronizarea multimedia și pentru mecanisme de adresare mai elocvente.

identic Exact la fel, după regulile pentru identitate ale **XML Namespaces.**

identificator general Numele unui element. Șirul asociat cu eticheta atunci când este folosit într-un document XML. `<!ELEMENT genericid ... > <genericid> stuff </genericid>`

ideogramă Un caracter dintr-o limbă care reprezintă un înțeles, și nu o pronunție.

IIOP (Internet Inter-ORB Protocol – protocolul Internet de comunicare între brokerii de cerere de obiecte) În **CORBA,** protocolul pentru comunicații între **ORB-uri** bazat pe TCP/IP. *Vezi* **GIOP.**

import În **XML** în general, a include date sau documente întregi din afara unui document în interiorul acestuia.

improvizație (kludge) O metodă de rezolvare a unei anumite sarcini, concepută la repezeală, incomodă și lipsită de eleganță.

includere În **XML** în general, acele lucruri care alcătuiesc conținutul unui element.

Infoset Schița de lucru W3C care descrie mulțimea de informații **XML.** *Vezi* **mulțime de informații.**

instanță O particularizare a unui obiect abstract.

instrucțiune În RDF, o expresie care numește o anumită **resursă** (subiect), o anumită **proprietate** (predicat sau atribut) și dă o valoare proprietății (complement sau **literal**) pentru acea resursă. *Vezi și* **metadate.**

instrucțiune de prelucrare Într-un **DTD,** o instrucțiune specială de forma `<?processor data ?>` care informează procesorul XML că trebuie să transfere datele incluse direct la un procesor extern.

Internet Rețeaua de calculatoare interconectate pe care se bazează **World Wide Web.** Internetul găzduiește multe servicii în afară de cele **Web,** cum ar fi e-mail, IRC, WAIS, Gopher și multe altele.

ISO 10646 Standardul internațional care definește reprezentări numerice unice pentru **simbolurile** utilizate în scrierea limbilor folosite în mod obișnuit în lume. Aproximativ echivalent cu **Unicode.**

înfrumusețare Prelucrarea unui fișier de cod astfel încât să respecte unele reguli de formatare care ușurează înțelegerea și editarea sa. *Vezi* **tipărire sugestivă.**

legătură externă Legătură care nu este conținută într-un document.

legătură extinsă O legătură conținută în afara unui **document XML.**

legătură in-line O legătură simplă, conținută chiar în document, și nu în afara acestuia ca în cazul **legăturii extinse.**

listă de atribute O listă de nume de atribute, cu tipuri de date și posibile valori prestabilite. *Vezi și* **ATTLIST**.

literal În **RDF**, un obiect, de obicei un șir de caractere. Conținutul unui literal nu este interpretat de RDF și poate să conțină marcaje XML suplimentare. Modelul RDF nu permite ca literalii să fie subiecte ale unei **instrucțiuni**.

locator În **XLink**, un obiect care fixează vizibil un capăt al unei legături într-un document.

marcaj Etichetele și comenzile de prelucrare care alcătuiesc structura documentului XML.

metadate O **instrucțiune** despre o **resursă**. *Vezi și* **RDF**.

metaforă Un principiu fundamental de organizare și operare care permite ca o aplicație să fie înțeleasă mai ușor în termenii conceptelor care există deja în mintea utilizatorului.

meta-limbaj Limbaj care se referă la structura documentului sau a elementelor, și nu la înțelesul lor literal.

model **Constrângerea** impusă conținutului unui element. *Vezi și* **tip de conținut**.

model de conținut Într-un **DTD**, șablonul care descrie **descendenții** legali ai unui anumit **nod**.

model de conținut any În **XML Schema**, un model de conținut care permite includerea oricărui conținut. *Vezi și* **model de conținut seq** și **model de conținut choice**.

model de conținut choice În **XML Schema**, un model de conținut care permite o singură alegere a unui element dintr-o listă. *Vezi și* **model de conținut seq** și **model de conținut any**.

model de conținut de grup all În **XML Schema**, un model de conținut care include totul. *Vezi și* **model de conținut de grup any**.

model de conținut de grup any În **XML Schema**, un model de conținut de grup care permite includerea oricărui conținut. *Vezi și* **model de conținut de grup seq** și **model de conținut de grup choice**.

model de conținut de grup choice În **XML Schema**, un model de conținut de grup care permite o singură alegere a unui element dintr-o listă. *Vezi și* **model de conținut de grup seq** și **model de conținut de grup any**.

model de conținut de grup seq În **XML Schema**, un model de conținut de grup care forțează toți membrii unei liste să apară secvențial în conținutul unui element. *Vezi și* **model de conținut de grup any** și **model de conținut de grup choice**.

model de conținut secvențial În **XML**, un model de conținut care forțează toți membrii unei liste să apară secvențial în conținutul unui element.

model de conținut seq În **XML Schema**, un model de conținut care forțează toți membrii unei liste să apară secvențial în conținutul unui element. *Vezi și* **model de conținut any** și **model de conținut choice**.

mulțime de informații În **XML** în general, o mulțime de date care descriu conținutul unui document XML. Formatul acestei mulțimi de date nu este definit și rămâne la latitudinea proiectantului. *Vezi* **Infoset**.

NCName Un **nume** fără calificativul de zonă de nume, așa cum este definit în **XML Namespaces**. Adică un nume care nu include un caracter două puncte.

nmtoken, nmtokens *Vezi* **reprezentare de nume**.

nod În general, un punct de ramificare sau o frunză într-un arbore al unui document. În **RDF**, o reprezentare a subiectului unei resurse sau un obiect **literal** în formatul grafului orientat; mai precis, un vârf într-un graf orientat, etichetat. *Vezi și* **XPath**, **arc** și alte intrări înrudite pe care le găsiți la aceste locații.

NOTATION O legătură la o entitate externă neanalizată.

notație În matematică, o metodă de a reda concepte matematice într-o formă concisă potrivită pentru afișare sau publicare. Notațiile fac adesea posibilă comunicarea unor idei extrem de complexe prin câteva simboluri sau prin aranjamente bidimensionale pe pagină ale unor simboluri preexistente.

nume Într-un **DTD**, un șir alcătuit dintr-o literă, o liniuță de subliniere sau un caracter două puncte, urmate de litere, cifre, liniuțe de subliniere, cratime, puncte, două puncte, caractere half colon, triangular colon, middle dot, accente de combinare, diacritice de combinare sau echivalente care nu sunt Latin-1. Utilizarea caracterelor două puncte este limitată la zone de nume și mecanisme interne **XML**. Utilizarea șirului inițial „xml" în orice combinație de tipuri de litere este interzisă, cu excepția numelor de atribute predefinite, cum ar fi `xml:lang`. Literele se referă la orice simbol ISO sau Unicode care funcționează ca o literă, silabă sau unitate ideografică cu înțeles, în orice limbă definită de pe glob. Analog, cifrele sunt simbolurile oricărui sistem de numerație.

obiect flux Un container XSL în care pot fi înglobate date.

ORB (Object Request Broker – broker de cerere de obiecte) În **CORBA**, infrastructura care permite obiectelor distribuite să comunice între ele.

origine *Vezi* **rădăcină**.

pădure *Vezi* **crâng**.

părinte **Predecesorul** imediat al unui **nod**.

Persistent Uniform Resource Locator (PURL – locator persistent și uniform de resurse) Un **URL** intermediat printr-un server de rezoluție care redirectează referințele spre locațiile lor actuale. Acesta asigură un nivel de persistență pentru resursele care ar putea să își schimbe adresele fizice pe Web.

PICS (The Platform for Internet Content Selection – platforma pentru selecția conținutului Internet) Dezvoltat inițial pentru a facilita controlul parental asupra accesului copiilor la siturile Web, acest sistem de etichetare de metadate suportă de asemenea și confidențialitatea în Internet și semnăturile digitale.

preambul Partea opțională introductivă a unui document XML, care conține orice combinație formată dintr-o declarație XML, o declarație DTD și instrucțiuni de prelucrare.

predicat În logică, un termen care indică o proprietate sau o relație. De asemenea, ceea ce se afirmă sau se infirmă într-o propoziție logică. Într-o propoziție, un verb sau o locuțiune verbală cu sau fără complement. În **RDF**, un atribut specific cu un sens definit care poate fi folosit pentru a descrie o **resursă** RDF. *Vezi* **proprietate**.

prelucrarea tranzacțiilor O tehnologie de procesare a datelor prin care serii întregi de operații trebuie să fie realizate ca un întreg sau pașii până la punctul eronat trebuie să fie parcurși în sens invers pentru a reface starea inițială a sistemului.

prezență În **XML Schema**, prezența unui element, a unui atribut sau a unei date opționale într-un document XML.

procesor XML Nivelul software care citește și interpretează documente XML, inclusiv orice declarații DTD necesare, verifică dacă documentele sunt construite corect, eventual le și validează și transferă conținutul lor analizat către o **aplicație**.

prolog Într-un DTD, acea parte a unui DTD XML sau a unui document care introduce și include declarația DOCTYPE. Este alcătuit din declarația XML, comentarii opționale și declarația tipului documentului.

proprietate În RDF, un alt nume pentru **predicat**. Un atribut specific cu un sens definit care poate fi folosit pentru a descrie o **resursă** RDF. *Vezi și* **arc, literal** și **metadate**.

prototip În **XML Schema**, o descriere de conținut pentru tipuri de elemente sau atribute, care poate fi refolosită și instanțiată sau declarată pentru un anumit element.

PURI *Vezi* **Persistent Uniform Resource Locator**.

racord În **CORBA**, modulul de comunicație care realizează interfața dintre nucleul **ORB** și implementarea **clientului**.

rafinare În **XML Schema**, procesul de dezvoltare din tipuri de date de bază pentru ca valorile primite de elementul care le include să fie și mai limitate.

rădăcină Baza arborelui unui document XML. Se mai numește și **origine**.

rădăcină a documentului *Vezi* **rădăcină**.

RDF (Resource Description Framework – cadrul descrierii resurselor) O metodă W3C de descriere a resurselor într-un **meta-limbaj** care nu transformă comportamentul nativ al resursei. RDF poate fi folosit pentru a eticheta documentele existente cu informații suplimentare, permițând o precizie mai bună și o granulozitate mai mare în căutări, fără a modifica aplicațiile existente. O generalizare în XML a unor cadre de metadate cheie, cum ar fi Dublin Core și alte descrieri ale obiectelor de date.

redare Afișare sau un alt tip de prezentare a documentului sau a unor părți ale documentului în maniera potrivită pentru un anumit dispozitiv. Redarea auditivă a documentelor care au fost create inițial pentru afișare vizuală le face mai accesibile persoanelor cu handicap de vedere, dar pentru a fi posibilă și utilă, este nevoie de o proiectare atentă.

referință caracter O entitate numerică de forma `&#xDDDD;` sau `&#xHHHH;` reprezentând un caracter din spațiul de caractere Unicode / ISO 8879 / ISO 10646. Entitățile zecimale sunt cele mai obișnuite, poate din experiența noastră cu ASCII. Însă echivalentele hexazecimale pot fi folosite fără conversii ca intrări în **Unicode Standard Version 3.0** și invers.

referință de tip În **XML Schema**, o referință la tipul de date al unui element.

referință element În **XML Schema**, o referință la un **element**, și nu la un **tip de date**.

referință entitate O referință la o **entitate** declarată anterior într-un **document XML**.

referință la un grup de modele În **XML Schema**, o referință la un grup de modele.

referință tip de date În **XML Schema**, o referință la tipul sau tipurile de date pe care le conține un **element**.

reprezentare de nume Într-un DTD, un șir ca un **nume**, dar fără restricții asupra primului caracter.

Resource Description Framework *Vezi* **RDF**.

resursă În general, orice unitate de informație adresabilă sau orice serviciu aflat oriunde în spațiul de informații format din calculatorul dumneavoastră, rețea sau World Wide Web. Orice lucru care poate fi adresat printr-un **URI**. În RDF, **nodul** care reprezintă subiectul unei **instrucțiuni** RDF. *Vezi și* **arc, literal** și **metadate**.

RUE (Reference to Undefined Entity – referință la o entitate nedefinită) Un element informație definit în **XML-Infoset**.

schemă O schemă definește în general structura și conținutul unui fișier XML. Limbaje de schemă particulare sunt **XML Schema, RDF** și **XML DTD**.

secțiune CDATA Informație care va fi transferată aplicației fără a fi analizată pentru a găsi marcaje, o prescurtare comodă pentru a include un întreg bloc de text într-o secvență escape fără a căuta fiecare caracter care, pentru a nu fi confundat cu un marcaj, ar trebui inclus într-o secvență escape. `<![CDATA[ anything ]]>`

secțiune condițională Într-un DTD, o metodă de a folosi o pereche de entități parametru pentru a comuta între structuri logice alternative ale DTD-ului. `<![INCLUDE[ anything ]]> <![IGNORE[ anything ]]>`

server Un obiect distribuit care primește cereri de la un **client** și răspunde la ele.

SGML (Standard Generalized Markup Language – limbajul de marcare standard generalizat) ISO 8879 este un limbaj de definire a marcajelor foarte complex, utilizat mai mult pentru proiecte foarte mari. **XML** încearcă să încorporeze cele mai valoroase caracteristici ale SGML într-un mod care să fie ușor de utilizat și implementat pe Web.

simbol Reprezentarea fizică a unei litere, cifre, **ideograme**, a unui caracter non-alfabetic sau a altui simbol în oricare dintre seturile de caractere **Unicode**.

sintaxă abstractă În **XML Schema**, o specificație formală a informației furnizate pentru fiecare declarație și definiție din limbajul de scheme. *Vezi și* **sintaxă concretă**.

sintaxă concretă În **XML Schema**, numele elementelor și atributelor folosite într-o schemă. *Vezi și* **sintaxă abstractă**.

spațiul de simboluri În **XML Schema**, domeniul de valabilitate al numelor folosite în definirea obiectelor.

strămoș În **XPath**, oricare dintre nodurile din secvența directă de noduri **părinte** de care depinde nodul context. *Vezi și* **el însuși**.

structură În **XML Schema**, domeniul modelelor de conținut și tipurilor de date posibile.

structură fizică Totalitatea unităților de stocare a datelor care formează un document XML și aranjarea lor.

structură logică Structura definită prin totalitatea declarațiilor XML dintr-un document, inclusiv cele definite ca entități interne și ca entități externe. Submulțimea externă a DTD-ului este un tip special de entitate externă.

submulțime externă a unui DTD Porțiunea unui DTD care este situată în afara unui document XML. *Vezi și* **submulțime internă a unui DTD**.

submulțime internă a unui DTD Porțiunea unui DTD conținută chiar în document. Orice procesor XML, indiferent dacă **validează** sau nu, trebuie să analizeze și să execute instrucțiunile din submulțimea internă, cu unele excepții.

tip Tipul de date al unui element.

tip de conținut Fie un **tip de date**, fie un **model de conținut**. *Vezi și* **conținut**.

tip de date Orice unitate de conținut care nu este un element. Date atomice sau terminale. Unul dintre cele două variante de **tip de conținut**.

tip de date atomic În **XML Schema**, un tip de date simplu, care nu este construit din alte tipuri de date. *Vezi și* **tip de date compus**.

tip de date compus În **XML Schema**, un tip de date complex, format din alte tipuri de date. *Vezi și* **tip de date atomic**.

tipărire sugestivă Prelucrarea unui fișier de cod astfel încât să respecte unele convenții de formatare care ușurează înțelegerea și editarea sa. De obicei, aceasta se realizează prin includerea fiecărei instrucțiuni pe o singură linie și indentarea fiecărei linii pentru a reflecta nivelul de imbricare. *Vezi* **înfrumusețare.**

topologie Conectivitatea permisă între noduri, ancore și legături. De exemplu, unu la unu, sau mai mulți la unu; unidirecțional sau bidirecțional și așa mai departe.

transcludere A prelucra și a include conținutul unei hiperlegături în corpul unui document. În HTML, imaginile, obiectele și foile de stil sunt transcluse în mod obișnuit. În XML, alte documente și porțiuni ale altor documente pot fi de asemenea redate în cadrul unui document.

triplet În **RDF**, reprezintă o **instrucțiune** alcătuită dintr-o **proprietate** (predicat), identificatorul **resursei** (subiect) și valoarea proprietății (complement sau **literal**), în această ordine. *Vezi și* **metadate.**

Unicode Standard Version 3.0 Ultima schiță a standardului Unicode publicată de Addison-Wesley, ISBN:0201616335. Este un instrument esențial pentru orice persoană din domeniul XML care este interesată de comunicațiile internaționale. *Vezi* **caracter.**

Unicode Standard internațional care definește reprezentări numerice unice pentru **simbolurile** folosite în scrierea limbilor cele mai utilizate din lume. Aproximativ echivalent cu **ISO 10646.**

URI (Uniform Resource Identifier – identificator de resurse uniform) O generalizare a metodelor URN și URL de adresare a informațiilor sau serviciilor, care include orice șir scurt ce se referă la o resursă aflată oriunde în spațiul de informații de pe Web.

URL (Uniform Resource Locator – locator de resurse uniform) RFC 1738 și RFC 1808. O metodă de adresare a informației pe Web prin intermediul unui identificator scurt care dă instrucțiuni explicite referitoare la locația unei informații pe Web.

URN (Uniform Resource Name – nume de resurse uniform) RFC 2141. Un mecanism care permite proprietarilor unei anumite zone de nume, de obicei, dar nu întotdeauna, o locație Web, să furnizeze o metodă de rezolvare a numelor în locații.

valid Un document **XML construit corect** care respectă și regulile definite în **DTD**-ul asociat.

validare Procesul prin care un document **XML** construit corect este analizat și testat pentru a asigura respectarea regulilor din **DTD**-ul care îl definește.

valoare fixă În **XML Schema**, conținut care nu variază ca valoare.

valoare prestabilită În **XML**, valoarea prestabilită pe care o primește un **atribut.**

variabilă independentă În matematică, o variabilă a cărei valoare poate să se modifice indiferent de valorile altor variabile. În mare, este variabila din membrul drept al unei ecuații. *Vezi* **variabilă legată.**

variabilă legată În matematică, o variabilă a cărei valoare depinde de valorile altor variabile. În mare, este variabila din membrul stâng al unei ecuații. *Vezi* **variabilă independentă.**

vocabular În **XML Schema**, o colecție preexistentă de tipuri de date și constrângeri, care poate fi refolosită pentru alte documente.

W3C (The World Wide Web Consortium – consorțiul World Wide Web) Un organism de standardizare alcătuit din reprezentanți ai furnizorilor și ai comunităților de utilizatori care definește și promulgă standarde legate de **World Wide Web.** `http://www.w3.org/`

Web **World Wide Web.** În mare, **Internetul.**

World Wide Web Rețeaua de calculatoare interconectate și mecanisme de transport care constituie mediul **Internet** interactiv cu hiperlegături, care ne este atât de familiar în ziua de azi.

XHTML (Extensible HTML 1.0 – HTML 1.0 extensibil) HTML 4.0 redefinit ca o aplicație a XML. Unul dintre subiectele acestei cărți.

XLink Limbajul de legare al **XML**, folosit pentru a defini **hiperlegături.**

XML (Extensible Markup Language 1.0 – limbajul de marcare extensibil) Principalul subiect al acestei cărți. XML permite descrierea serială a unui arbore de date abstract, ierarhic.

XML Namespaces Un mecanism prin care elemente din **DTD**-uri diferite pot partaja un **document XML** fără a exista conflicte de nume.

xml:lang Un atribut folosit pentru a declara limba folosită în interiorul unui element dat sau al unei entități. Declarația respectă RFC 1766 pentru etichetele de identificare a limbii, ISO 639 pentru codurile numelor de limbi și ISO 3166 pentru codurile numelor de țări. `<name xml:lang="xx">`. Acest atribut este predefinit, dar nu este declarat în nici un element.

xml:space Un atribut folosit pentru a declara intenția de a păstra spațiile albe din interiorul unor elemente care, altfel, le-ar ignora. `<!ATTLIST name xml:space (default|preserve) 'preserve'>`. Acest atribut este predefinit, dar nu este declarat în nici un element.

XPath Un limbaj compact, proiectat pentru a adresa porțiuni ale unui document XML sau înrudit cu XML într-o manieră lipsită de ambiguități. Este o parte a **XLink, XPointer, XSL, XSLT** și a altor recomandări W3C.

XPointer Locatorul extins al unui document identificat printr-un **URI** care permite identificarea precisă a poziției într-un document sau într-o porțiune selectată dintr-un document. *Vezi și* **XPath.**

XSL (Extensible Stylesheet Language – limbajul extins pentru foi de stil) O reformulare și o extindere a **CSS2** ca **document XML** cu adăugiri semnificative care îi permit să se refere la o gamă largă de documente și la un model de pagină mai puternic. Mai simplu, proprietățile și valorile în stil CSS2 sunt convertite în atribute și valori în stil XML, cu adăugiri importante la funcționalitatea **CSS2.**

XSLT (Extensible Stylesheet Language Transformations – limbajul extins pentru transformarea foilor de stil) O facilitate subordonată a **XSL** care permite transformarea unui document **XML** în altul.

zonă de nume Un identificator arbitrar, folosit pentru a evita conflictele de nume între diferite DTD-uri, sau un echivalent prescurtat. În unele standarde legate de XML, cum ar fi **XML Schema**, o prescurtare a rădăcinii unui arbore în care se poate găsi un fișier aplicabil unui document (cum ar fi o schemă). Zonele de nume sunt uneori folosite ca și comutatoare interne pentru a indica unui analizor XML sau unui agent utilizator că trebuie realizate anumite tipuri de prelucrări. *Vezi și* **domeniu de valabilitate.**

Index

SIMBOLURI

*** (asterisc) EBNF, 645-646**
// şiruri comentariu JavaScript, 145
+ (plus) EBNF, 646
? (semn de întrebare) EBNF, 645-646
*** selector, 287**
(<!—)şiruri comentariu SGML/XML, 144
4DOM, sit Web cu instrumente, 682-683
4XPath, sit Web cu instrumente, 683
4XSLT, sit Web cu instrumente, 683
#CDATA, text, 144
#PCDATA, text, 144

A

a priori, 99
a, element (SMIL), 507, 520-521
<a>, etichetă (HTML), 180
abrevieri (XPath), 171-174
  exemple, 174-175
  lista, 175-177
Academia Sinica Computing Centre, sit Web, 623
acceptare, construirea DTD-urilor, 112
accesibilitate Web, 582-584
  aspecte legale, 587-588
  avantaje comerciale, 586-587
  depanare
    cerinţele clientului, 609
    plângerile utilizatorilor, 608
  dispozitive de ieşire Braille, 595-596
  exemplu de document XML, 600
    aspectul paginii, 602-603, 607-608
    cadre, 601-602
    pagina de intrare XHTML a librăriei, 603-604
    pagina de navigare, 605-606
    textul pentru indexul librăriei, 600-601
  limbi străine, 584-586
  redare auditivă
    editare vocală, 599
    foi de stil auditive, 596-599
    medii ţintă multiple, 599
  SMIL (Synchronized Multimedia Integration Language), 536
    legende, 528-530
    naraţiuni, 528-530
    probleme legate de dispozitivele mobile, 534-535
    secvenţe audio descriptive, 531-534
  tehnici de afişare pentru vedere slabă, 588-589
    cadre, 591-592
    conţinut multimedia plug-in, 593
    fişiere PDF(Portable Document Format), 592
    fundale, 589
    imagini, 589-590
    legături, 590-591
    medii de redare JavaScript, 592
    notaţii image, 589-590
    tabele, 593-595
    text, 589
accesibilitate, SMIL (Synchronized Multimedia Integration Language)
  legende, 528-530
  naraţiuni, 528-530
  probleme legate de dispozitivele mobile, 534-535
  secvenţe audio descriptive, 531-534
Acord, 615
ACTION, apelarea CGI, 386-387
Active Server Pages (pagini server active). *Vezi şi* ASP
ActiveX, 405
  DLL-uri, creare în Visual Basic, 416-417
acţiunea new (XLink), 197
acţiunea parse (XLink), 197
acţiunea replace (XLink), 197
adăugare
  atribute, 94
  elemente, 95
    DTD-uri, 103-104
  ERP, la sisteme de întreprindere, 456-457
Adequate.com, 472
adresare absolută (XPointer), 201
adresare numai prin nume (XPointer), 184
adresare relativă (XPointer), 200-201
adresare XPointeri
  adresare absolută, 201
  adresare relativă, 200-201
adunarea datelor (DTD-uri), 113-114
adunarea numelor elementelor, 114-115
Aegis.com, sit Web, 258
afişarea paginilor
  tehnici pentru vedere slabă, 588-589
    cadre, 591-592
    fişiere PDF (Potrtable Document Format), 592
    fundale, 589
    imagini, 589-590
    legături, 590-591
    medii de redare JavaScript, 592
    notaţii de imagini, 589-590
    plug-in-uri cu conţinut multimedia, 593
    tabele, 593-595
    text, 589

AIML (Astronomical Instrument Markup Language), 580, 616
alegerea parolelor, 362-363
alfabete, coregrafia XML, 632
Amaya
  browser, 569
AML (Astronomical Markup Language), 580, 616
analiza traficului, 368
analizoare
  DOM, 390
  Lark, 682
  MSXML, 406
    downloads, 406
  SAX, 229, 681
  TclXML, 682
  tipuri de elemente, 52-53
  XP, 681
analizor SAX, 681
  sit Web, 681
analizorul Java XML XML4J
  instalare, 382
  sit Web, 382
analizorul Lark, 682
  sit Web, 682
ancore (HTML), 180
  creare, 180-181
  probleme cu, 181
ANZLIC (Australia New Zealand Land Information Council), 616
apelare, CGI
  ACTION, 386-387
  GET, 386-387
  POST, 386-387
API-uri, 226
aplicaţia cu suport XML Breeze Commerce Studio, 686
  sit Web, 686
aplicaţie cu suport XML Amaya, 686
  sit Web, 686
aplicaţii, 58
  CML, 59
  comerciale, 610-611
  cu suport XML
    Amaya, 686
    Breeze Commerce Studio, 686
    Mathematica, 685
    MathType, 685
    WebEQ, 685
  dezvoltare, 423-424
  GedML, 59
  Health Level-7 (HL7), 58
  ICE (Information and Content Exchange), 59
  Java CGI, învelişuri Perl, 383-384
  MathML, 59
  MISTI (Missile Industry Supply-chain Transaction Infrastructures), 59
  moştenite, 425-426
  Real Estate Transaction Standard (RETS), 59
  RosettaNet, 59
  SAE J2008, 59
  SMIL, 59
  standarde, 59
  verificatoare ortografice, 423-424
aplicaţii comerciale, 610-611
aplicaţii cu suport XML
  Amaya, 686
  Breeze Commerce Studio, 686
  Mathematica, 685
  Math Type, 685
  WebEQ, 685
aplicaţii existente, 58-59
  CML, 59
  GedML, 59
  Health Level-7 (HL7), 58
  ICE (Information and Content Exchange), 59
  MathML, 59
  MISTI (Missile Industry Supply-chain Transaction Infrastructures), 59
  Real Estate Transaction Standard (RETS), 59
  RosettaNet, 59
  SAE J2008, 59
  SMIL, 59
  standarde, 59
aplicaţii moştenite, 425-426
APnP (Appliance Plug and Play), 439
APPEL, sit Web, 620
Apple Quicktime, 504
Appliance Plug and Play (APnP), 439
Application (Re)Engineering de Amjad Umar, 426
arbori
  arborele documentului DOM Level 2, 212-215
  crânguri, 212
  filtre, 214-215
  indicatoare whatToShow, 213-214
  vederea logică, controlul, 213
argumente, scheme XML, 143-145
arhitectură (XML), 15
  descendenţi, 16-17
  excludere, 15-16
asistenţi, profiluri utilizator (confidenţialitate), 343-345

ASP (Active Server Pages), 405-412
Content Type, 412
depanarea codului Visual Basic, 421-422
obiect cerere, 408-409
obiect răspuns, 409-411
obiecte COM, 415-416, 420-421
testare, 411, 420-421
Associating Style Sheets with XML documents Version 1.0, sit Web, 620
asterisc (*), EBNF, 645-646
Astronomical Instrument Markup Language (AIML), 580, 616
Astronomical Markup Language (AML), 580, 616
atacuri
hacker-i, 354
Internet Worm, 352-353
Spoofing, 353-354
AtGuard, 359
atribut abstract
elemente obiecte multimedia, 510-511
elementul par, 508
elementul seq, 509
atribut actuate (XLink), 190
atribut extended (XLink), 189
atribut NOTATIONS, 648-649
atribute
atribute ale interfeței traversal (DOM), 217
atribute de interfață pentru nod (DOM), 216-217
ca selectori, 286
Cascading Style Sheets, 713
CDATA, 77
codificare, 646
diferențe între DOM Level 1 și DOM Level 2, 209-210
DTD, modificare 91
elemente, adăugare, 94-95
entități, 78-79
enumerate, 81
ID, 77
IDREF, 77-78
locale, 152-156
minimizat HTML, 484
NMTOKEN, 79
NOTATION, 648-649
scheme XML
constrângere, 162-163
definire, 158
validarea tipurilor de date, 152-156
SMIL
atribute de testare a sistemului, 519-520
elemente obiecte multimedia, 510-511
elementul a, 520-521
elementul anchor, 521
elementul head, 504
elementul layout, 504-505
elementul meta, 506
elementul par, 508-509
elementul region, 505-506
elementul root-layout, 506
elementul seq, 509-510
elementul switch, 518-519
special (HTML), 485
specificarea datelor prestabilite, 76
standalone, 646
condiții de validitate, 647
XLink, 189-191
actuate, 190
arc, 190
auto, 190
extended, 189
from, 190
href, 189
locator, 189
new, 190
none, 190
parse, 190
replace, 190
resource, 190
role, 191
show, 190
simple, 189
title, 190, 191
to, 190
type, 189
user, 190
xmlns, 130-133, 676
xsi:targetNamespace, 149
atribute enumerate, 81
atribute event-value
elemente obiecte multimedia, 510-511
elementul par, 508
atribute minimizate (HTML), 484
atribute simple (XLink), 189
atribute speciale (HTML), 485
atributul alt (elemente obiecte multimedia), 510
atributul arc (XLink), 190
atributul author
elemente obiecte multimedia, 510-511
elementul par, 508
elementul seq, 509
atributul auto (XLink), 190
atributul background-color (element region), 506
atributul begin
elemente obiecte multimedia, 510-511
elementul a, 521
elementul par, 508
elementul seq, 509
atributul clip-begin (elemente obiecte multimedia), 511
atributul clip-end (elemente obiecte multimedia), 511
atributul coords (elementul a), 521
atributul copyright
elemente obiecte multimedia, 510-511
elementul par, 508
elementul seq, 509
atributul dur
elemente obiecte multimedia, 510-511
elementul par, 508
elementul seq, 509
atributul end
elemente obiecte multimedia, 510-511
elementul a, 521
elementul par, 508
elementul seq, 509
atributul endsync (elementul par), 508
atributul fill (elemente obiecte multimedia), 511
atributul fit (elementul region), 506
atributul from (XLink), 190
atributul height (elementul region), 506
atributul href (XLink), 189
atributul id
elemente obiecte multimedia, 510-511
elementul a, 521
elementul par, 508
elementul seq, 509
atributul left (elementul region), 506
atributul locale, 152-156
atributul longdesc (elemente Media Object), 511
atributul new (XLink), 190
atributul none (XLink), 190
atributul parse (XLink), 190
atributul region
elemente obiecte multimedia, 510-511
elementul par, 508
elementul seq, 509
atributul repeat
elementul par, 508
elementul seq, 509
atributul replace (XLink), 190
atributul resource (XLink), 190
atributul role (XLink), 191
atributul show
elementul a, 521
elementul par, 508
elementul seq, 509
XLink, 190
atributul skip-content
elementul a, 521
elementul region, 505-506
atributul src (elemente obiecte multimedia), 511
atributul standalone
constrângeri de validitate, 647
DTD-uri, 646
atributul system-bitrate, 519
elemente obiecte multimedia, 510-511
elementul par, 508
elementul seq, 510
elementul switch, 518-519
atributul system-captions, 520
elemente obiecte multimedia, 510-511
elementul par, 508
elementul seq, 510
elementul switch, 518-519
atributul system-language, 520
elemente obiecte multimedia, 510-511
elementul par, 508
elementul seq, 510
elementul switch, 518-519
atributul system-overdub-or-caption, 520
elemente obiecte multimedia, 510-511
elementul par, 508
elementul seq, 510
elementul switch, 518-519
atributul system-required, 520
elemente obiecte multimedia, 511
elementul par, 508
elementul seq, 510
elementul switch, 518-519
atributul system-screen-depth, 520
elemente obiecte multimedia, 511
elementul par, 509
elementul seq, 510
elementul switch, 518-519
atributul system-screen-size, 520
elemente obiecte multimedia, 511
elementul par, 509
elementul seq, 510
elementul switch, 518-519
atributul title
elemente obiect multimedia, 511
elementul a, 521
elementul par, 509
elementul region, 505-506
elementul seq, 510
XLink, 189-191
atributul to (XLink), 190
atributul top (element regiune), 506

atributul type
elemente obiect multimedia, 511
elementul layout, 504-505
XLink, 189
atributul user (XLink), 190
atributul width (element region), 506
atributul xmlns, 130-133, 676
atributul xsi:noNameSchemaLocation, 149
atributul xsi:targetNamespace, 149
atributul z-index (element region), 506
ATTLIST
declarații, 677
redefinire, 94-95
audio
CSS, 739-743
Tipuri de medii SMIL, 525-526
VoiceXML, 526-528
VoxML, 526
Australia New Zealand Land Information Council (ANZLIC), 616
avantajele XML
dezvoltarea Web, 15
flexibilitatea machetelor, 14
independența de platformă, 14
precizie, 14
proiectare orientată spre obiecte, 15
validarea structurii documentului, 14
axe (XPath), 169-171

B

BAAN
ERP, furnizori actuali, 454-455
sit Web, 454
Bank Internet Payment System (BIPS), 616
banner-e, riscuri de securitate, 369
baze de date server-side, profiluri utilizator, 335-336
baze de date XML. *Vezi și* dbXML
baze de date, server-side, stocarea informațiilor profilurilor de utilizator, 335-336
BBBOnline, sit Web (profiluri de utilizatori), 342
Better Business Bureau (profiluri de utilizatori), sit Web, 342
Bibliomania, sit Web, 128
binary, grupul de lucru XML Schema al W3C, 161
Bio.Perl, sit Web, 616
Bioinformatic Sequence Markup Language (BSML), 616
BIOML (BIOpolymer Markup Language), 616
BIOpolymer Markup Language (BIOML), 616
BIPS (Bank Internet Payment System), 616
BizTalk, sit Web, 59
BlackICE Defender de la Network ICE, 359-361
BlackICE Defender, 359-361
blindate
DTD-uri, 125-126
comentarii, 123-124
elemente, 124
nume de entități, 124
boolean, grupul de lucru XML Schema al W3C, 159
broker de obiecte COM
cerere, 426-427
componente Microsoft, combinare, 431
comunicarea cu brokerul de obiecte CORBA, 427
DEC, 430
broker de obiecte RMI, cerere, 426-427
brokeri de obiecte, 426-431. *Vezi și* ORB-uri
modele, standarde, 428
COM+, 431
nori, 425
broker-i de profiluri, confidențialitate, 336
brokerul de obiecte CORBA
cerere, 426-427
citirea resurselor, 429
comunicarea cu brokerul de obiecte COM, 427
depanare, 429
DII, 429
IDL, 429
ORB-uri, 429
browsere
Amaya, 569
auditive
abilitate, 297-298
distincția între vorbitori, 298
setarea proprietăților, 298
browserul Mozilla, foi de stil, 264
browserul Opera
foi de stil, 262, 273, 323
CSS, 283-284
cu suport XML, 285
HTML, reparații pe loc, 286
implementarea CSS, 285
incompatibilitate, depanare XHTML, 496-498
InDelv, foi de stil, 267-268
Microsoft Internet Explorer, foi de stil, 272
Mozilla, 285
MultiDoc Pro, 279
Netscape, foi de stil, 272
Opera, 285
notația cu acolade, 296
suport pentru foi de stil, 269-279, 323, 746-747
suportul CSS, compatibilitate simulată, 286
vizualizarea unei pagini XML, 312
BrowserHawk, 258
brwsere cu suport XML
Amaya, 680
InDelv XML Client, 680
Internet Explorer 5 (Microsoft), 680
JUMBO, 680
Mozilla, 680
Panorama SGML Viewer, 680
Plume, 681
browserul cu suport XML Amaya, 680
sit Web, 680
browserul InDelv, foi de stil, 267-268
BSML (Bioinformatic Sequence Markup Language), 616
bug-uri (CSS), 747
business to business (B2B) sisteme de tranzacții, 60
byte, grupul de lucru XML Schema al W3C, 160

C

C2net, 367
cadre, tehnici de afișare pentru vedere slabă, 591-592
Cafe Con Leche: XML News and Resources, sit Web, 624
calculatoare mobile, SMIL (Synchronized Multimedia Integration Language), 534-535
calea de includere a claselor, servlet-uri, 404
CALS (Continuous Acquisition and Lifecycle Support system), 612
DTD-uri, sit Web, 613
câmpuri de intrare cu format liber, scheme XML, 165
Canonical XML, sit Web, 621
caractere
marcaje XML, EBNF, 646
XML construit corect, 83-84
caractere de marcare (XML), EBNF, 645-646
caracteristica traversal (DOM), 211
caracteristicile CSS hard, 286
Cascading Actions Systems, coregrafia XML, 633. *Vezi și* Sistemul de acțiune în cascadă Netscape
Cascading Style Sheets level 1. *Vezi și* CSS1
Cascading Style Sheets level 2. *Vezi și* CSS2
Cascading Style Sheets. *Vezi și* CSS
caseta de dialog References, 419
casete (CSS), 723-729
casete cu citat evidențiat, 293
casete de dialog, referințe, 419
casete de text mobile simulate, 295
Cayman Systems, 355
căi de locație relative, suport XPath, 170-171
căi de locație, suportul XPath pentru acestea, 170-171
căi de locații absolute
suport XPath, 170
căutare pe Web
căutări fără context, 463-464
PICS, 470-472
RDF, 462-463
CDATA, 75
atribute, 76
conținut, agenți utilizator HTML, 284
prelucrare, 52
secțiuni, 651
secvențe escape, 82
Centre de recherche en droit public (CRDP), 616
century, grupul de lucru XML Schema al W3C, 161
cercetare criptografică, 369
cerere
broker de obiecte, 426-427
COM, 426-427
CORBA, 426-427
RMI, 427
XMLTP, 427
servelt-uri XML, 399-402
cererea GET, 392
apelarea CGI, 386-387
cereri GET, 392
CERT (Computer Emergency Response Team), 369
CFML (Cold Fusion Modeling Language), 616
CGI
ACTION, apelare, 386-387
GET, apelare, 386-387
Java
aplicații, învelișuri Perl, 383-384
execuție, 383
POST, apelare, 386-387
probleme, scalabilitate, 391-392
Chami, sit Web, 676
Chami.com, sit Web, 37

Chemical Markup Language. *Vezi și* CML
Chinese XML Now!, sit Web, 623
Chrome (Microsoft), coregrafia XML, 634. *Vezi și* Microsoft Chrome
CIDS (Component Information Dictionary Specification), 616
cifrare. *Vezi și* criptare
Cisco Systems, 355
CiTEC, 279
citire de resurse, brokerul de obiecte CORBA, 429
cititorul de ecran JAWS, 298
clasa AttributeImpl, 233-238
clasa AttributeListImpl, 234-235
clasa AttributesImpl, 235-236
clasa DefaultHandler, 234-237
clasa InputSource, 234, 243-244
clasa LocatorImpl, 234, 244
clasa NamespaceSupport, 234-244
clasa ParseFactory, 235
clasa ParserAdapter, 234, 245
clasa ParserFactory, 245
clasa XmlDocument, obținere, 397
clasa XMLFilterImpl, 235
clasa XMLReaderAdapter, 235, 252-253
clasa XMLReaderFactory, 235, 253
clase, 254
  AttributeImpl, 234
  AttributeListImpl, 234-235
  AttributesImpl, 235-236
  CSS, 290
  DefaultHandler, 234-237
  HandlerBase, 234, 243
  InputSource, 234, 243-244
  instanțiere, 254
  LocatorImpl, 234, 244
  moștenire, 254-255
  NamespaceSupport, 234, 244
  ParserAdapter, 234, 245
  ParserFactory, 235, 245
  pseudo, 289
  XMLFilterImpl, 235
  XMLReaderAdapter, 235, 252-253
  XMLReaderFactory, 235, 253
clase interfață (RMI), 437-438
CMIS (Configuration Management Information System), 612
CML (Chemical Markup Language), 59, 540, 576-580, 616
cod CGI (scriere)
  extragerea parametrilor, 388-389
  parametri, 390-391
  tipul conținutului, 388
cod sursă, testarea validității, 101-102
codare URL, inversare, 389-390
codificări de caractere, 43-45
codificări, 43-45
  atribute, DTD-uri, 646
  URL, inversare, 389-390
Coins: tightly coupled JavaBeans and XML elements, 616
Cold Fusion Modeling Language (CFML), 616
color (CSS), 719
Colossus, 364
COM (Component Object Model), 430-431
  ActiveX, 405
  DCOM, 431-432
    MTS, 434
    UUID-uri Registry, 432
  dezavantaje, 431
  ORPC, 432
COM and CORBA Side by Side, 426
comentarii, 284-285, 651
  Cascading Style Sheets, 713
  DTD-uri blindate, 123-125
comenzi
  comenzi CSS, 745-746
  foi de stil, 284-285
  procesare, 650-651
Commerce XML (cXML), 616
CommerceNet Home Page, sit Web, 616
Common Markup for micropayment per-fee-links, sit Web, 621
Common Object Request Broker Architecture. *Vezi și* CORBA
compatibilitate (DTD-uri), zone de nume, 126-127
compatibilitate cu bug-urile anterioare (CSS), 286
Component Information Dictionary Specification (CIDS), 616
Component Object Model. *Vezi și* COM
componente Microsoft, combinarea broker-ilor de obiecte COM, 431
Computer Emergency Response Team (CERT), 369
comunicare
  între brokeri de obiecte, 427-428
  ORB-uri, protocoale, 425
comunicări între procese
  XMLTP, 434
    depanare, 435
    situri Web, 435
conceptul Generic Coding, 28
confidențialitate
  criptare, 363-364
    asimetrică, 364-367
    criptare puternică, restaurare, 367
    criptare TCP/IP, 368
    instrumente de criptare a fișierelor, 367
    simetrică, 364
  firewall-uri
    definire, 355
    HTTP (Hypertext Transfer Protocol), 358-359
    porturi binecunoscute, 356-358
    porturi dinamice, 358
    porturi înregistrate, 358
    produse firewall existente, 359-361
    servere proxy, 356
  funcții hash, 375-376
  inițiative de securitate și confidențialitate, 368-370
  P3P, 376
  parole, alegere, 362-363
  profiluri utilizator, 333-334
    asistenți de confidențialitate, 343-345
    asistenți, 343-345
    brokeri de confidențialitate, 336
    ID-uri/parole utilizator, 336-337
    OPS/P3P, 341
    P3P, implementate cu Java, 342
    P3P, politici de examinare, 340
    P3P, preferințe de confidențialitate, 337-338
    P3P, schimb de politici de confidențialitate, 339-340
    P3P, secvența evenimentelor, 338-339
    utilizarea bazelor de date server-side, 335-336
    utilizarea cookie-urilor, 334-335
  rutere, 349-350
    avantaje, 354-355
    cheltuieli, 354
    stiva protocolului TCP/IP, 350-352
  semnături digitale, 370
    SMDL (Signed Markup Description Language), 371-374
    XML semnat, 370-371
  SSL (Secure Sockets Layer), 374-375
  vulnerabilitatea datelor, 347-349
configurare
  IIS, 385-386
  PWS, 385-386
Configuration Management Information System. *Vezi și* CMIS, 612
conlucrarea între aptitudinile și posibilitățile schemelor XML, 148
ConSeal Private Desktop, 359
Consorțiul World Wide Web. *Vezi și* W3C
constrângerea conținutului element/atribut, scheme XML, 162-163
constrângeri
  construire corectă, 41-42, 640
  validitate, atributul standalone, 646
constructori, 253-254. *Vezi și* metode
construire corectă, 38-41, 52-53
  constrângeri, 41-42, 640
containeri, attlist, 75
Continuous Acquisition and Life-cycle Support system. *Vezi și* CALS
contoare (CSS), 732
conținut any, 75
conținut ordonat, entități, 71
conținut terminal, 73-74
conținutul selectării, entități, 71-72
convenții
  denumire, 114-115
  elemente, 485-486
convenții de marcare ebraice, 27
convenții de marcare în ebraica veche, 27
convenții de marcare în latina veche, 27-28
convenții de marcare în latina veche, 27-28
conversie
  SGML DTD la XML, 97
  XHTML de la HTML, 489-496
conversii, sistem ERP, 456-457
cookie-uri, profiluri de utilizatori, 334-335
Coordinating Committee for Intercontinent Research, 369
coordonarea tipurilor de mediu SMIL cu instrumente multimedia, 528
CORBA (Common Object Request Broker Architecture), 402-404
  XML, 404
coregrafia (XML), 626-627
  alfabete multiple, 632
  Comportamente Microsoft DHTML, 633
  depanare, 636-637
  fluxuri de date multiple, 634-635
  limbi naturale, 628
  limbile Microsoft Internet Explorer, 631
  limbile Mozilla, 629-631
  limbile Opera, 632
  localizare, 628-629
  mai multe limbi, 628
  mai multe medii, 632
  mai mulți utilizatori, 635
  medii interactive, 632-633
  Microsoft Chrome, 634
  Sistemul de acțiune în cascadă Netscape, 633

suport pentru limbi, 629-631
utilizatori, 627-628
viitorul HTML, 636
VRML, 633-634
Corpus Legis Project, sit Web, 616
CRDP (Centre de recherche en droit public), 616
creare
comenzi de foi de stil, 288
declarații attlist, 75
documente, 39
selectori (CSS), 286
situri Web (CSS), 283-284
crearea de DTD-uri
acceptare, 112
implementare, 112
iterație, 112
personalizat, 111
proiectare, 112
testare, 112
criptare, 354, 363-364
asimetrică, 364-367
criptare puternică, restaurare, 367
criptare TCP/IP, 368
instrumente de criptare a fișierelor, 367
simetrică, 364
criptare asimetrică, 364-367
criptare simetrică, 364
criptologie. *Vezi și* criptare
CSS (Cascading Style Sheets - foi de stil în cascadă), 259, 283-284
atribute
căutare, 713
valori, 713
beneficii, 283-284
bug-uri, 747
caracteristici hard, 286
clase, 290-291
comanda important, 715
comentarii, 284-285, 713
comenzi, 745-746
compatibilitate cu bug-urile anterioare, 286
compatibilitatea CSS1 și CSS2, 709
contoare, 732
conținerea în HTML, 712
convenții de formatare a textului, 711-712
creare
selector, 286
CSS2
abilități media, 297-298
selectarea elementelor, 286
CSS3
elemente mobile paginate, 293
proprietăți comportamentale, 302
suport și emulare pentru copii tipărite, 293
depanare, 281, 303
diferențe între versiuni, 290, 293
efecte de text, 293
elemente, 286
exemplu, 709-710
formatare grafică, 285-286
grupare, 712
grupuri de medii, 716-717
implementări ale browserelor, 285
importanța, 745
interfața cu utilizatorul, 737-739
limite, 285
listing, 259, 269-272
list-style-type, 730
marcaje pentru citate, 731-732
Microsoft, 286
moștenire, 712
numerotare, 732
ordinea cascadelor, 715-716
pagini, 732-734
perechi proprietate/valoare, 290
proprietatea content, 731
proprietăți de clasificare, 729-731
proprietăți pentru casete, 723-729
proprietăți pentru culoare, 719
proprietăți pentru font, 717-719
proprietăți pentru fundal, 719-720
proprietăți pentru text, 721-723
proprietăți, 287
pseudo-clase, 289, 714-715
pseudo-elemente, 289, 714-715
recomandări, 710
referințe, 748
regula @import, 715
regula @media, 716
reguli, 287
selectori, 284-288
clase, 712
contextuali, 713
copii, 713
descendenți, 713
frați adiacenți, 713
ID, 713
servicii de validare, 710, 748
sintaxa, 284-285
stiluri, 285
sunet, 739-743
suport în browsere, 269-274, 283-284, 746-747
compatibilitate simulată, 286
Netscape, 303
tabele, 734-737
tipuri de medii, 291-292, 716
extindere, 302
proprietăți auditive, 298-299
sensibilitatea la tipul literelor, 292
unități de măsură, 743-744
URL-uri, 743-744
URN-uri, 743-744
utilizarea cu XSL, 279-280
validatorul pentru foi de stil TopStyle, 297
white space, 730
CSS1 (Cascading Style Sheets level 1), 258-262, 620
compatibilitate cu CSS2, 709
listing, 260-261
sit Web, 620
CSS2 (Cascading Style Sheets level 2), 258-264, 620, 748-749
compatibilitate cu CSS1, 709
listing, 262-263
sit Web, 620
cunoștințele din domeniu, proiectarea DTD-urilor, 113-114
Customer Support Consortium, sit Web, 616
cuvinte cheie (EBNF), sensibilitate la tipul literelor, 646
cXML (Commerce XML), 616
Cyberkit, 501

D

Data Documentation Initiative, sit Web, 616
Data Interchange Standards Association (DISA), 617
date
CDATA, 75, 77
coregrafia XML, fluxuri de date, 634-635
entități, 75-79
ID, 75, 77
IDREF, 75, 78
IDREFS, 75
implementare, 114-115
instrucțiuni de prelucrare, 82
limbaje pentru asigurarea datelor, 442
literal, 63
metadate, 460-461
Combinarea schemelor de metadate, 474-475
Dublin Core, 460-461, 464-470, 473
PICS, 460-461, 470-472
RDF, 460-465, 473-480
vizualizare, 460-461
NMTOKEN, 76, 79
notații, 75-76
prestabilite, 76
resurse, gestionare, 443-444
scheme XML
definire, 158-161
extindere, 162
stocare, 442
RTF-uri, 442
tipuri, validare, 152-156
date caracter analizate, 73
date caracter, 47-48
date literale, 63
date prestabilite, specificare, 76
date, grupul de lucru XML Schema al W3C, 160
David Raggett's HTML Tidy instrumente de conversie XML, 682
sit Web, 683
dbXML (baze de date XML), 445
situri Web, 445
DCE/RPC (Distributed Computing Environment's Remote Procedure Call), 432
situri Web, 432
DCOM (Distributed COM - COM distribuit), 431
modelul de obiecte COM, 431-432
MTS, 434
UUID-uri Registry, 432
DDML (Document Definition Markup Language), 167
DDML/Xschema, sit Web, 616
DEC (Digital Equipment Corporation), 431
brokerul de obiecte COM, 431
decimals, grupul de lucru XML Schema al W3C, 159
declarare
attlist, 62-63, 75
elemente, 71
entități, 62, 66-67
neanalizate, 70
parametru, 67
notații, 64-65
zone de nume, 130-131
identificate prin contexte non-standard, 140-141
identificate prin ISBN, 139
identificate prin ISSN, 139
identificate prin URN, 139
în limbi non-europene, 137-138
prefixate, 131-133
declararea attlist, 62-63, 75
declarații, 43
ATTLLIST, 677
codificări de caractere, 43-45
documente autonome, 45-46
ELEMENT, 677
ENTITY, 677

NOTATION, 648-649
fișiere GIF, 649
versiuni, specificare, 646
declarații de confidențialitate, sit Web (profiluri utilizator, situri Web), 343
declarații ELEMENT, 677
declarații ENTITY, 677
declarații NOTATION, 648-649
fișiere GIF, 648-649
definire
atribute, scheme XML, 158
clasele interfeței RMI, 437-438
elemente, scheme XML, 157
ERP, 448
modele de conținut, scheme XML, 162
rădăcini elemente, 116118
tipuri de date, scheme XML, 158-161
zone de nume, 135-136
definiție, scheme XML, validarea tipurilor de date, 152
denumire
convenții, 115
DTD-uri, compatibilitate, 127-128
entități, 53-54
etichete, 34-35
ID-uri, 291
depanare
accesibilitate
cerințele clientului, 609
plângerile utilizatorilor, 608
foi de stil, 303
CSS, 281
XSL, 282
incompatibilitate browser XHTML, 496, 498
instrumente pentru DTD-uri, 123
probleme de securitate
exemplu, 376-377
instrumente sistem Windows, 377-378
RDF, 479-480
SAX (API simplu pentru XML), 256
SMIL (Synchronized Multimedia Integration Language), 537
XLink, 204
legături întrerupte, 204
sintaxa de legare, 204
XML, 636-637
XPointer-i, 204
legături întrerupte, 204
sintaxa de legare, 204
XSL, 329
depozite de date, integrarea XML, 441
descendenți (XML), 16-17
Desktop Management Interface (DMI), 617
destructori, 253-254. *Vezi și* metode
determinare
valori de timp în SMIL, 512-513
începutul dorit al elementelor, 516
începutul efectiv al elementelor, 516
începutul implicit al elementelor, 513-514
sfârșitul dorit al elementelor, 515
sfârșitul efectiv al elementelor, 516-517
sfârșitul implicit al elementelor, 514-515
determinarea începutului elementelor (SMIL)
începutul dorit al elementelor, 516
începutul efectiv al elementelor, 516
începutul implicit al elementelor, 513-514
determinarea sfârșitului elementelor (SMIL)
sfârșitul dorit al elementelor, 515
sfârșitul efectiv al elementelor, 516-517
sfârșitul implicit al elementelor, 514-515
developerWORKS, IBM, sit Web, 623
dezvoltare, 15
instrumente
Cocoon, 681
Multidoc Pro Publisher, 681
Near & Far Designer, 681
Python and XML Processing, 681
Visual XML, 681
XMetaL 1.2, 681
XML <PRO>, 681
XML Spy, 681
XML Testbed, 681
XMLwriter, 681
dezvoltare Web, 15
dezvoltarea
aplicații, 423-424
servlet-uri, 392
DHTML (Dynamic HTML), 633. *Vezi și* Microsoft DHTML
coregrafia XML, comportamente, 633
Digital Equipment Corporation. *Vezi și* DEC
Digital Receipt Infrastructure Initiative, sit Web, 617
DII (Dynamic Invocation Interface)
Brokerul de obiecte CORBA, 429
Dynamic Invocation Interface, 429
DISA (Data Interchange Standards Association), 617
sit Web, 617
dispozitive
hand-held, 301
ieșire, medii, 292
telefoane mobile și pagere, afișaje, 301
dispozitive de ieșire Braille, 595-596
dispozitive de redare
auditive, 298
hand-held, 301
miniaturale, 301
dispozitive de redare auditive
CSS2, 297
distincția între vorbitori, 298
setarea proprietăților, 298
dispozitive hand-held, 301
Distributed COM. *Vezi și* DCOM
Distributed Computing Environment's Remote Procedure Call. *Vezi și* DCE/RPC, 432
Distributed InterNet Applications Architecture (DNA), 407-408
DLL-uri, creare cu Visual Basic, 416-417
DMI (Desktop Management Interface), 617
sit Web, 617
DMTF Common Information Model, sit Web, 617
DMTF Web-based Enterprise Management (WBEM), sit Web, 617
DNA (Distributed InterNet Applications Architecture), 407-408, 432
DocBook (Documentation Made Portable), 614-615
DTD-uri, 59, 614-615
etichete, 131
sit Web, 614-615
Document Definition Markup Language. *Vezi și* DDML
Document Object Model (DOM). *Vezi și* DOM
Document Style Semantics and Specification Language (DSSSL - limbajul de semantică și specificare a stilului documentului), 268
Document Type Declaration (declarația tipului documentului), 46-47
Document Type Definitions. *Vezi și* DTD-uri
Documentation Made Portable. *Vezi și* DocBook
documente
arbori
arborele documentului în DOM Level 2, 212-215
filtre, 214-215
indicatorii whatToShow, 213-214
perspectiva logică, control, 213-215
construire corectă, 38-41, 52-53
constrângeri, 41-42
creare, 39
date analizate, 38
date caracter, 47-48
date neanalizate, 38
declarații, 43
codificări de caractere, 43-45
documente autonome, 45-46
Document Type Declaration, 46-47
documente numai pentru citire
legare la, 200
documente SMIL (Synchronized Multimedia Integration Language)
antet, 504-506
corp, 505-507
creare, 502-503
identificarea unei poziții în, 291
marcaje, 48
atribute, 54-56
entități, 49-50, 53-54
etichete, 48-51
poziție în
urmărire, 172-174
profiluri, 490
avantaje, 490
proiectare orientată spre obiecte, 15
tipuri, 61
ținte, 291
validarea structurii, 14
vizualizare, 210
XML, redare, 109
documente autonome, 45, 46
documente numai pentru citire, legare, 200
Doczilla, 279-280
DOM (Document Object Model), 206-208
analizoare, 390
arborele documentului nivelul 2, 212-215
atribute, 209-210
atributele interfeței node, 216-217
capricii, 223-224
caracteristica Traversal, 211
Cascading Style Sheets (CSS), 210
diferențele între nivelele 1 și 2, 209-212
DOM la nivel de nod, 216-217
DOM orientat spre obiecte, 218-223
domeniu, 211
evenimente, 211
filtre, 214-215
foi de stil, 210
HTML dinamic, 208
indicatorii whatToShow, 213-214
interfețe extinse, 219
interfețe fundamentale, 218-219
interfețe HTML, 210
limitele actuale, 208
metode, 206-210, 220-223
metodele interfeței node, 216-217
perspectiva la nivel de nod, 207-208

perspectiva logică, control, 213-215
perspectiva orientată spre obiecte, 206
perspective, 210
structură, 223
suport pentru limbaje de programare, 224-225
tratarea excepțiilor, 224
DOM la nivel de nod, 216-217
atributele interfeței node, 216-217
atributele interfeței traversal, 217
metodele interfeței node, 216-217
DOM Level 1, sit Web, 621
DOM Level 2, sit Web, 621
double, grupul de lucru XML Schema al W3C, 159
downloads
Internet Explorer 5, 406
JWSDK, 382-383
MSXML, 406
Personal Web Server (PWS), 406
DSSSL, 268
DTD local, indicare, 85
DTD.com, sit Web, 624
DTD-ul CALS, 612
DTD-ul XHTML 1.01 Tranzitoriu, listing, 653-675
DTD-uri (definițiile tipului documentului), 24, 30-31, 61-62, 612
acceptare, creare, 112
atributul encoding, 646
atributul standalone, 646
avantaje, 32-33
blindat, 125-126
CALS, 612
CDATA, 75, 77
comentarii, blindat, 123-124
compatibilitate
denumire, 128
zone de nume, 126-127
conținut any, 75
conținut terminal, 73-74
conținut vid, 74
design
creare, 112
principii, 120-121
dezavantaje, 33-34
DocBook, 59, 614-615
Document Type Declaration, 46-47
ELEMENT, 650
elemente
adăugare, 103-104
blindat, 124
declarare, 70
tipuri, 52-53
entități, 66-67, 75-76, 78-79
generale, 69
neanalizate, 69-70
parametru, 67-68, 69
entități parametru, modificare, 95-97
erori de recursivitate, 122
extensibil, creare, 105
extern, 84
furnizarea de valori locale, 108-109
indicare, 84-85
extindere, 93-94, 102
fragmente, 84-87
HTML, încorporarea elementelor, 99-101
ID, 75, 77
identificatorul de zonă de nume xmlns, 99
IDREFS, 75, 78
implementare, construire, 112-117
indicatori de apariții, 72-73
instrucțiuni DOCTYPE, 647
instrumente, depanare, 123
iterație, construire, 112
limbaje, 58-59
literale, 63
local, indicare, 85
localizare, 127-128
logică circulară, 122
MARC, 615
marcaje condiționale, 90
modificare, 104-110
modificare, 90-91
modificare, forță brută, 104
modular, 106-107
creare, 105-107
multiple, cu zone de nume, 129-130
NMTOKEN, 76
notații, 64-65, 75-76
nume de entități, blindare, 124
particularizat, creare, 111-112
privat public, ISO-12083, 613-614
proiectare
adunarea datelor, 113-114
colectarea cunoștințelor din domeniu, 113-114
scopuri, 114
prologuri, 646
SGML, 612-613
convertirea la XML, 97
sintaxa, 62-63
submulțimea externă, 67-68
submulțimea internă, 67
submulțimi, 30-31
și performanță, 30-31
TEI, 614
test
construire, 112
testare, 118-120
timp de încărcare, 30-31
XHTML 1.01 Tranzitoriu, listing, 653-675
XHTML
etichete ancoră, 676
extensibilitate, 677
sensibilitate la tipul literelor, 675
zone de nume, 676
XML construit corect, 83-84
zone de nume XML, 106
zone de nume, 105
DTD-uri extensibile, creare, 105
DTD-uri externe, 84
furnizarea de valori locale, 108-109
indicare, 84-85
submulțime, 67-68
DTD-uri modificate, 104-105
DTD-uri modulare, 106-107
creare, 105-107
DTD-uri personalizate, creare, 111-112
DTD-uri private publice, ISO-12083, 613-614
DTD-uri SGML, 484, 612-613 *Vezi și* **HTML XML, 97**
Dublin Core, 460-473
Dynamic Invocation Interface. *Vezi și* **DII**

E

E:lang:(), 287
EAD (Encoded Archival Description), 617
sit Web, 617
EBNF (forma Backus-Naur extinsă), 31-32
* (asterisc), 645-646
? (semn de întrebare), 645-646
+ (plus), 646
caracterele de marcare XML, 645-646
cuvinte cheie, sensibilitate la tipul literelor, 646
EBNF (forma Backus-Naur extinsă), producții, 639
gramatica XML, 640-645
ECIX (Electronic Component Information Exchange), 617
sit Web, 617
editare
fișiere XML, 135-136
voci (redare auditivă), 599
editare vocală (redare auditivă), 599
editoare
Amaya, 679
Emacs, 679
Microsoft XMLNotePad, 679
XML Spy, 679
XMLwriter, 679
editorul Amaya, 679
sit Web, 679
editorul de ecuații WebEQ, 575
editorul Emacs, 679
editorul GRiNS de la Oratrix, 523
editorul GRiNS, 523
editorul Microsoft XMLNotePad, 679
editorul XMLNotePad, 679
Educom Instructional Management Systems Project (IMS), 617
EEMA EDI/EC Work Group: UN-XML, sit Web, 617
Electronic Component Information Exchange (ECIX), 617
Electronic Frontier Foundation, 369
Electronic Privacy Information Center, 369
ELEMENT (DTD-uri), 650
elemente
adăugare, 95
valori pentru proprietăți, 288
atribute, adăugare, 94-95
ca selectori, 286
CONSEQUENCE (P3P), 339-340
conținut any, 75
conținut empty, 74
conținut terminal, 73-74
convenții, 485-486
CSS, 286
CSS2, 286-287
DATA (P3P), 339-340
declarare, 70
DISCLOSURE (P3P), 339
DISPUTES (P3P), 339
DTD XHTML, încorporare, 99-101
DTD-uri
adăugare, 103-104
blindare, 124
modificare, 91
excluderi
HTML, 485
SGML, 675
ID-uri XML, 288
imbricare, 121
liste, paranteze, 116
nume, adunare, 114-115
perechi proprietate/valoare, 290
plasare, 99
POLICY (P3P), 339
pseudo, 289
PURPOSE (P3P), 339-340
rădăcină, definire, 115-117
RECIPIENT (P3P), 339
Repetare, 72-73

RETENTION (P3P), 339-340
Scheme XML
conținut de constrângere, 162-163
conținut de validare, 163
definire, 157
STATEMENT (P3P), 339
tipuri, 52-53
vid, 650
XLink
elemente arc, 193
elemente extinse, 192-193
elemente locator, 193
elemente resource, 193-194
elemente simple. 198-199
elemente title, 194
elemente (SMIL)
a, 507, 520-521
anchor, 521
animation, 507
audio, 507
body, 507-508
determinare valori de timp, 513
începutul dorit al elementelor, 516
începutul efectiv al elementelor, 516
începutul implicit al elementelor, 513-514
sfârșitul dorit al elementelor, 515
sfârșitul efectiv al elementelor, 516-517
sfârșitul implicit al elementelor, 514-515
elemente obiecte multimedia, 510-511
head, 504
hiperlegare, 520-521
img, 507
layout, 504-505
meta, 506
par, 507-509
ref, 508
region, 505-506
root-layout, 506
seq, 508-510
smil, 504
switch, 508, 518-519
text, 508
textstream, 508
video, 508
elemente arc (XLink), 193
elemente cu conținut repetat, 72-73
elemente de animație (SMIL), 507
elemente de plasare, 99
elemente imbricate (HTML), mecanisme de recuperare, 486
elemente legale, 623. *Vezi și* accesibilitate
elemente mobile paginate, 293
elemente obiecte multimedia (SMIL), 510-511
elemente paragraf, valori de proprietăți, 288
elemente resource (XLink), 193-194
elemente simple (XLink), 192
elemente title (XLink), 194
elementul anchor (SMIL), 521
elementul audio (SMIL), 507
elementul body (SMIL), 507-508
elementul CONSEQUENCE (P3P), 339-340
elementul DATA (P3P), 339-340
elementul DISCLOSURE (P3P), 339
elementul DISPUTES (P3P), 339
elementul head (SMIL), 504
elementul img (SMIL), 507
elementul meta (SMIL), 506
elementul par (SMIL), 507-509
elementul POLICY (P3P), 339
elementul PURPOSE (P3P), 339-340
elementul RECIPIENT (P3P), 339
elementul ref (SMIL), 508
elementul region (SMIL), 505-506
elementul RETENTION (P3P), 339-340
elementul root-layout (SMIL), 506
elementul seq (SMIL), 508, 509-510
elementul smil (SMIL), 504
elementul STATEMENT (P3P), 339
DOCTYPE, 647
instrucțiuni Dim (obiecte COM), 419
elementul switch (SMIL), 508, 518-519
elementul text (SMIL), 508
elementul textstream (SMIL), 508
Encoded Archival Description (EAD), 617
Enterprise Resource Planning. *Vezi și* ERP
entitate
Grupul de lucru XML Schema al W3C, 161
nume, DTD-uri blindate, 124
entități, 49-50, 75-76
analizate recursiv, 122
atribute, 78-79
caractere XHTML
sortate alfabetic, 698-703
sortate numeric, 703-708
conținut ordonat, 71
declarare, 62-63, 66-67
denumire, 53-54
entități neanalizate, 56-58
fragmente DTD, 85-86
generale, 69
Grupul de lucru XML Schema al W3C, 161
ISO
caractere matematice, grecești și simbolice, 693-698
caractere speciale, 692-693
Latin, 688-692
neanalizate, 69-70
parametru, 67-69, 87
evitare în text arbitrar, 125-126
extindere, 97-98, 118-119
modificare DTD-uri, 95-97
Recomandarea XML 1.0 a W3C, 647-648
selectare de conținut, 71-72
sintaxă, 66
tabelul W3C, 88
entități analizate recursiv, 122
entități analizate, recursivitate, 122
entități caracter ISO matematice, grecești și simbolice (setul de caractere XHTML), 693-698
entități caracter ISO speciale (setul de caractere XHTML), 692-693
entități generale, 69
entități ISO Latin-1 (setul de caractere XHTML), 688-692
entități neanalizate, 56-58, 69-70. *Vezi și* notații
declarare, 70
entități parametru, 67-69, 87
DTD-uri
marcaj condițional, 90
modificare, 95-97
extindere, 97-98, 118-119
tabelul W3C, 88
text arbitrar, evitare, 125-126
EPOC-32 sistem standard de operare nepatentat, 301
Equation Editor, 539
erori
condiții, verificare (obiecte COM), 418
HTML, 286
recursive (DTD-uri), 122
scheme XML, limite, 164-165
erori recursive (DTD-uri), 122
ERP (Enterprise Resource Planning), 448
adăugarea la sistemele întreprinderii, 456-457
conversii de sisteme, 457
definite, 448
furnizori actuali, 454-455
BAAN, 454, 456-457
ERP Hub, 454
Manex, 454
Mapics, 454
Oracle, 454-455
PeopleSoft, 456
PivotPoint, 454
SAP R/3, 454, 455-456
Hub
furnizori actuali, 454-455
sit Web, 455
integrarea XML în, 449-450
limbaje comune, 450-451
între sisteme de întreprindere, 452
mediul care se modernizează, 452-453
migrare
rezultate, 454
strategii, 453, 456
rezumat, 448-449
eticheta <applet> (HTML), 181
eticheta <area> (HTML), 180
eticheta <base> (HTML), 180
eticheta <blockquote> (HTML), 181
eticheta <body> (HTML), 181
eticheta <del> (HTML), 181
eticheta <form> (HTML), 181
eticheta <frame> (HTML), 180
eticheta <head> (HTML), 181
eticheta <iframe> (HTML), 180
eticheta <img> (HTML), 180
eticheta <input> (HTML), 181
eticheta <ins> (HTML), 181
eticheta <link> (HTML), 180
eticheta <object> (HTML), 181
eticheta <param> (HTML), 181
eticheta <q> (HTML), 181
eticheta <script> (HTML), 181
etichete
ancoră (DTD-uri XHTML), 676
de încheiere, 50-51
de închidere, 37
DocBook, 131
etichete de conținut, 50
etichete de pornire, 50-51
etichete vide, 50-51
HTML
<a>, 180
<area>, 180
<base>, 180
<blockquote>, 181
<frame>, 180
<head>, 181
<iframe>, 180
<img>, 180
<ins>, 181
<param>, 181
<q>, 181
de închidere, 484
limbă, 485
opțional, 485
imbricare, 35, 50

numire, 35
ordine, 21-23
virtual, 287
XML construit corect, 83-84
etichete ancoră
DTD-uri XHTML, 676
legare, XLink, 183
etichete conținut, 50
etichete de încheiere, 37, 50-51
HTML, 484-485
etichete de marcare. ***Vezi și*** **etichete**
etichete de pornire, 50-51
etichete opționale (HTML), 485
etichete virtuale, 287
eTrust, 368
EUROMATH Project, sit Web, 617
European XML/EDI Pilot Project, sit Web, 617
evaluarea siturilor Web, 470-472
evitarea entităților parametru, text arbitrar, 125-126
excepția SAXException, 234, 246-247
excepția SAXNotRecognizedException, 234, 247
excepția SAXNotSupportedException, 234, 247
excepția SAXParseException, 234, 247
excepții
SAXException, 234, 246-247
SAXNotRecognizedException, 234, 247
SAXNotSupportedException, 234, 247
SAXParseException, 234, 247
Exchange Table Model, sit Web, 617
excluderi recursive, 17
excluderi, 15-16
elemente (SGML), 675
execuție
Java CGI, 383
programe Java CGI, 404
HTML, 386-387
servlet-uri, 404
expresii, MathML, 554
extensibilitate, 17
DTD-uri XHTML, 676
XHTML, 486-487
preprocesare specială, 487-488
probleme de imbricare, 487
Extensibility XML Authority, instrumente, 683
sit Web, 683
Extensible Forms Description Language (XFDL), 620
Extensible Hypertext Markup Language. ***Vezi și*** **XHTML**
Extensible Markup Language (XML), transformare în HTML pe server, 412-415
Extensible Stylesheet Language. *Vezi și* XSL
extensii (XPointer), 202-203
extindere
DTD-uri, 93-94, 102-105
tipuri de date, scheme XML, 162
extinderea entităților parametru, 97-98, 118-119
extins
elemente (XLink), 192-193
interfețe (DOM), 218-219
legături, 186-189
transcludere, 185, 196

F

facilități server-side (ActiveX), 405
familii de voci, 299
Federal Logistics Information System. ***Vezi și*** **FLIS**
filtre (DOM Level 2), 214-215
finalizoare, 253
firewall-uri, 355-356
definire, 355
HTTP (Hypertext Transfer Protocol), 358-359
porturi binecunoscute, 356-358
porturi dinamice, 358
porturi înregistrate, 358
produse firewall existente, 359-361
servere proxy, 356
SOAP, 438-439
fișier catalog, 651-652
fișier DLL, creare (obiecte COM), 420
fișiere
catalog, 651-652
fișier DLL, creare (obiecte COM), 420
GIF, declarații NOTATION, 648-649
PDF (Portable Document Format), tehnici de afișare pentru vedere slabă, 592
XML, editare 138
fișiere GIF, declarații NOTATION, 648-649
fișiere PDF (Portable Document Format)
tehnici de afișare pentru vedere slabă, 592
fișiere text multimedia, transcludere, 185
FLIS (Federal Logistics Information System), 612
float, grupul de lucru XML Schema al W3C, 159
fo (formatting objects, obiecte de formatare), 311-312
proprietăți de formatare, 313-320

foi de stil, 258
alegere, 259, 265, 268
arbori de transformare, 326-328
browsere, 262
browserul Mozilla, 264
browserul Opera, 262, 273
InDelv, 267-268
Microsoft Internet Explorer, 272
Netscape, 272
cascade, 283-284
casetă de text mobilă, 294
comentarii, 284-285, 713
comenzi, 745-746
CSS, 259
atribute, 713
bug-uri, 747
comanda important, 715
comentarii, 284-285, 713
comenzi, 745-746
compatibilitatea CSS1 și CSS2, 709
contoare, 732
conținerea în HTML, 712
convenții de formatare a textului, 711-712
depanare, 281
exemplu, 709-710
grupare, 712
grupuri de medii, 716-717
importanța, 745
interfața cu utilizatorul, 737-739
listing, 259-260
list-style-type, 730
marcaje pentru citate, 731-732
moștenire, 712
numerotare, 732
ordinea cascadelor, 715-716
pagini, 732-734
proprietatea content, 731
proprietăți de clasificare, 729-731
proprietăți pentru casete, 723-729
proprietăți pentru culoare, 719
proprietăți pentru font, 717-719
proprietăți pentru fundal, 719-720
proprietăți pentru text, 721-723
pseudo-clase, 714-715
pseudo-elemente, 714-715
recomandări, 710
referințe, 748
regula @import, 715
regula @media, 716
selectori, 712-713
servicii de validare, 710, 748
sunet, 739-743
suport în browsere, 269-274, 746-747
tabele, 734-737
tipuri de medii, 716
unități de măsură, 743-744
URL-uri, 743-744
URN-uri, 743-744
utilizarea cu XSL, 279-280
white space, 730
CSS1, 258-262
listing, 260-261
CSS2, 258-262, 264, 748-749
listing, 262-263
depanare, 282, 303, 329
DSSSL, 268
efecte de text, 293
externe, 284-285
foi de stil auditive, 596-599
formatare grafică, 285-286
formatting objects (fo, obiecte de formatare), 311-312
implementări ale browserelor, 285
instrumentul TopStyle, 273-274
limite, 285
listing, 266-267, 274-277
listinguri, 323-328
Microsoft, 322
notația cu punct, 286
notațiile cu punct, 286
numerotare, 322
proprietăți sigure, 285-286
pseudo-clase, 289
pseudo-elemente, 289
selectori, 284-288
sortare, 321
stiluri, 285
suport în browsere, 269-274, 283-284, 746-747
șabloane, 305-311, 320
testare, 274
tipuri de medii
extindere, 302
proprietăți auditive, 298-299
utilizarea cu CSS, 279-280
validatorul pentru foi de stil TopStyle, 297
XHTML și XML, 277-279
XSL (Extensible Stylesheet Language), 25-26, 259, 264-268, 304
foi de stil auditive, 596-599, 739-743
foi de stil Explorer, 272
<FONT FACE> Considered Harmful, sit Web, 258
fonturi (CSS), 717-719
FOP
instrumente, 682
sit Web, 682
forma arhitecturală Hy-Time, 166
Forma Backus-Naur extinsă. ***Vezi și*** **EBNF**
formatting objects (fo), obiecte de formatare, 311-312
proprietăți de formatare, 313-320

forme arhitecturale, Hy-Time, 166
formulare
  metoda doGet(), trimitere, 393
  servleţ-uri, prelucrare, 394-395
Fortify, 367
forţă brută, modificarea DTD-urilor, 104
fragmente (DTD-uri), 84-87
frecvenţa vorbirii, 299
frecvenţă (auditiv), unităţi de măsură, 300
Frontier 5 and XML, sit Web, 617
funcţia concat() (XPath), 178
funcţia contains() (XPath), 178
funcţia count() (XPath), 177
funcţia end-point(), 203
funcţia here(), 203
funcţia id() (XPath), 177
funcţia id(), 203
funcţia last() (XPath), 177
funcţia local-name() (XPath), 177
funcţia local-name(), 170
funcţia name() (XPath), 177
funcţia namespace-uri(), 170
  XPath, 177
funcţia origin(), 203
funcţia position() (XPath), 177
funcţia start-point(), 203
funcţia starts-with() (XPath), 178
funcţia string() (XPath), 178
funcţia string-length() (XPath), 178
funcţia substring() (XPath), 178
funcţia substring-after() (XPath), 178
funcţia substring-before() (XPath), 178
funcţia unique(), 203
funcţii
  end-point(), 203
  funcţii hash, 375-376
  here(), 203
  id(), 203
  local-name(), 170
  namespace-uri(), 170
  origin(), 203
  start-point(), 203
  unique(), 203
funcţii (XPath), 177
  boolean(), 179
  ceiling(), 179
  concat(), 178
  contains(), 178
  count(), 177
  false(), 179
  floor(), 179
  id(), 177
  lang(), 179
  last(), 177
  local-name(), 177
  name(), 177
  namespace-uri(), 177
  normalize-space(), 178
  not(), 179
  number(), 179
  position(), 177
  round(), 179
  starts-with(), 178
  string(), 178
  string-length(), 178
  substring(), 178
  substring-after(), 178
  substring-before(), 178
  sum(), 179
  translate(), 178
  true(), 179
funcţii booleene (XPath)
  boolean(), 179
  false(), 179
  lang(), 179
  not(), 179
  true(), 179
funcţii booleene (XPointer), 203
funcţii pentru locaţii absolute (XPointer), 203
  end-point(), 203
  here(), 203
  id(), 203
  origin(), 203
  start-point(), 203
funcţii pentru mulţimi de noduri (XPath)
  count(), 177
  id(), 177
  last(), 177
  local-name(), 177
  name(), 177
  namespace-uri(), 177
  position(), 177
funcţii principale (XPath), 177
  boolean(), 179
  ceiling(), 179
  concat(), 178
  contains(), 178
  count(), 177
  false(), 179
  floor(), 179
  id(), 177
  lang(), 179
  last(), 177
  local-name(), 177
  name(), 177
  namespace-uri(), 177
  normalize-space(), 178
  not(), 179
  number(), 179
  position(), 177
  round(), 179
  starts-with(), 178
  string(), 178
  string-length(), 178
  substring(), 178
  substring-after(), 178
  substring-before(), 178
  sum(), 179
  translate(), 178
  true(), 179
funcţii referitoare la numere (XPath)
  ceiling(), 179
  floor(), 179
  number(), 179
  round(), 179
  sum(), 179
funcţii string (XPath)
  concat(), 178
  contains(), 178
  normalize-space(), 178
  starts-with(), 178
  string(), 178
  string-length(), 178
  substring(), 178
  substring-after(), 178
  substring-before(), 178
  translate(), 178
fundale
  CSS, 719-720
  tehnici de afişare pentru vedere slabă, 588-589
furnizarea de valori locale (DTD-uri externe), 108-109
furnizori ERP actuali, 454-455
furnizori, ERP actual, 454-455

G

GCA (Graphics Communications Association)
GEDCOM (GEnealogical Data COMmunication), 59
GedML, 59
GedML: Genealogical Data in XML, sit Web, 617
GEnealogical Data COMmunication (GEDCOM), 59
Generalized Markup Language (GML), 28
Geometry Technologies, 575
Georgia State University Electronic Court Filling Project, sit Web, 617
germană, publicarea de documente în, 584-586
gestionarea evenimentelor, SAX (Simple API for XML), 230-232
gestiunea resurselor de date, 443
  Poet, 443-444
gestiunea resurselor de date, Poet, 443-444
Global Positioning System (GPS), 501
GML (Generalized Markup Language), 28
GNOME (Gnu Network Object Model Environment), 624
  sit Web, 624
Gnu Network Object Model Environment (GNOME), 624
GPS (Global Positioning System), 501
grade centezimale (grad), 300
grade sexagesimale (deg, degrees), 300
grafică (CSS), 285-286
gramatică (XML) listingul EBNF, 640-645
granulozitatea datelor (XML), 15-16
  descendenţi, 16-17
  excludere, 15-16
Graphical User Interface. *Vezi şi* GUI
Graphics Communications Association (GCA), 617
grupare, Cascading Style Sheets, 712
grupul de lucru XML Schema
  binary, 161
  boolean, 159
  byte, 160
  century, 161
  date, 160
  decimal, 159
  double, 159
  entities, 161
  entity, 161
  float, 159
  id, 161
  idref, 161
  idrefs, 161
  int (întregi), 159
  integer, 159
  language, 161
  long (întregi), 159
  month, 160
  name, 161
  ncname, 162
  negativeInteger, 160
  nmtoken, 161
  nmtokens, 161

nonNegativeInteger, 160
nonPositiveInteger, 160
notation, 161
notations, 161
positiveInteger, 160
qname, 161
recurringDate, 160
recurringDay, 160
recurringDuration, 160
short (întregi), 160
string, 159
time, 160
timeDuration, 160
timeInstant, 160
timePeriod, 160
unsignedByte, 160
unsignedInt, 160
unsignedLong, 160
unsignedShort, 160
uri-reference, 161
year, 160
grupuri din domeniu, 615. ***Vezi și*** **DTD-uri**
Acord, 615
AIML, 616
AML, 616
ANZLIC, 616
Bio.Perl, 616
BIOML, 616
BIPS, 616
BSML, 616
CFML, 616
CIDS, 616
CML, 616
Coins: Tightly Coupled JavaBeans and XML Elements, 616
Commerce XML, 616
CommerceNet Home Page, 616
Corpus Legis Project, 616
CRDP, 616
Customer Support Consortium, 616
Data Documentation Initiative, 616
DDML/XSchema, 616
Digital Receipt Infrastructure Initiative, 617
DISA, 617
DMI, 617
DMTF Common Information Model, 617
DMTF Web-based Enterprise Management (WBEM), 617
EAD, 617
ECIX, 617
EEMA EDI/EC Work Group: UN-XML, 617
EUROMATH Project, 617
European XML/EDI Pilot Project, 617
Exchange Table Model, 617
Frontier 5 and XML, 617
GCA, 617
GedML: Genealogical Data in XML, 617
Georgia State University Electronic Court Filling Project, 617
Health Level Seven, 617-618
HR-XML Consortium, 618
HTTP Distribution and Replication Protocol (DRP), 618
IEEESTD V3.0 DTD, 618
IMS, 617
Information & Content Exchange (ICE), 618
Interactive Financial Exchange (IFX), 618
JSML, 618
Meta Content Framework (MCF), 618
Newspaper Association of America (NAA), 618
Ontology Markup Language, 618
Open Content Syndication (OCS), 618
Open Software Description Format (OSD), 618
Open Trading Protocol (OTP), 618
OpenMLS, Real Estate DTD Design, 618
OpenTag, 618
Platform for Internet Content Selection (PICS), 618
Process Interchange Format XML (PIF-XML), 618
Rich Site Summary (RSS), 618
RosettaNet, 618
Sable, Speech Synthesis Markup Language, 619
SAE J2008, Automotive Industry Standard, 619
SIM SGML Database Technology, 619
SmartX, 619
SOAP, 620
SSML, 619
TalkML, 619
Telecommunications Interchange Markup (TCIF/IPI), 619
Theological Markup Language (ThML), 619
Translation Memory eXchange, 619
UIML, 619
UML eXchange Format (UXF), 619
Virtual Hyperglossary (VHG), 619
VoiceXML, 619
Weather Observation Markup Format (OMF), 619
Web Standards Project (WSP), 619
Web-based Distributed Authoring and Versioning (WEBDAV), 619
Wireless Markup Language (WML), 619
XFDL, 620
XLF Initiative, 620
XML Court Interface (XCI), 620
XML NITF, 620
XML Schema Site, 619
XML/EDI Group, 620
XML/EDI Repository, 620
XMLNews, 620
XML-RPC Home Page, 620
GUI (Graphical User Interface - interfață cu utilizatorul grafică), 425
mesaje, ORB-uri, 425
Obiecte, 425-426

H

HandlerBase
clasă, 234, 243-244
handler, 230
hardware, Palm Pilots, 301
hash, 375-376
Health Level-7. ***Vezi și*** **HL7**
Henter-Joyce, sit Web, 297-298
Hertz (Hz), 300
hiperlegături
SMIL (Synchronized Multimedia Integration Language), 520
elementul a, 520-521
elementul anchor, 521
tehnici de afișare pentru vedere slabă, 590-591
HL7 (Health Level-7), 58
sit Web, 617
HR-XML Consortium, sit Web, 618
HTML (Hypertext Markup Language), 19, 34-35, 37, 481-482. ***Vezi și*** **SGML, XHTML**
agenți utilizator, conținut CDATA, 284-285
ancore, 180
creare, 180-181
probleme cu, 181
atribute minimizate, 484
atribute speciale, 485
browsere, rezolvarea erorilor pe loc, 286
Cascading Style Sheets, 712
conversia la XHTML, 489-496
conversia la XML, 290
coregrafia XML, viitorul, 636
documente, ținte, 291
elemente, adăugarea de valori la proprietați, 288
etichete
<a>, 180
<area>, 180
<base>, 180
<blockquote>, 181
<frame>, 180
<head>, 181
<iframe>, 180
<img>, 180
<ins>, 181
<param>, 181
<q>, 181
etichete de încheiere, 484
etichete de limbă, 485
etichete opționale, 485
excluderea de elemente, 485-486
fișiere, Hypatia, 492-495
HTML dinamic, Document Object Model (DOM), 208
ID-uri, 291
legături, 180
creare, 180-181
probleme cu, 181
limitările, 13
mecanisme de revenire
elemente imbricate incorect, 486
instrucțiuni de procesare, 486
spații albe, 486
minimizări, 481-482
Pretty HTML, 483
programul Tidy, 482-483
transformarea în XHTML, 482-483
programe Java CGI, execuție, 386-387
script-uri și foi de stil, 485
sensibilitatea la tipul literelor, 484
sintaxa de clasă cu punct, 288
suportul pentru seturi de caractere, 485
translatarea XML în, 412-415
validare XHTML, 484-488, 490
valori de atribute, 484
XML, transformare, 397-399
HTML 4.0, sit Web, 621
HTML 4.01, sit Web, 621
HTML dinamic (DOM), 208
HTML-Kit / Tidy (XHTML), 676
HTML-Kit from Chami.com
instrumente de conversie XML, 682
sit Web, 682
HTML-Kit, 491-494
HTTP (Hypertext Transfer Protocol), 358-359
dezavantaje, 402
informații
transfer, 402
HTTP Distribution and Replication Protocol (DRP), sit Web, 618
Hypatia, fișiere HTML, 492-495
Hypertext Markup Language. ***Vezi și*** **HTML**
Hypertext Transfer Protocol. Vezi HTTP

I

IBM alphaWorks Tool Suite
instrumente, 683-685
ICE (Information and Content Exchange), 59
id
atribut
elemente de machetare, 504-505
grupul de lucru XML Schema al W3C, 161
ID, 75
atribute, 77
identificare
scheme XML, compromisuri, 147-148
zone de nume, 133-134
în atribute, 134-135
în elemente, 134-135
identificatori zonă de nume (xmlns), 99
identificatorul de zonă de nume xmlns (DTD-uri), 99
IDL (Interface Definition Language - limbajul de definire a interfeței)
brokerul de obiecte CORBA, 403, 429
Interface Definition Language, 423-424
obiecte, 424-425
IDL CORBA, 403
IDREF, 75
atribute, 77-78
idref, grupul de lucru XML Schema al W3C, 161
profiluri de utilizator, 336-337
adrese de e-mail, 336-337
ID-uri
HTML, 291
XML, 291
IEEESTD V3.0 DTD, sit Web, 618
ieșire
arbori XML, crearea (obiecte COM), 418-419
copii tipărite, 293
dispozitive
medii, 292
SMIL (Synchronized Multimedia Integration Language), 536
tipuri de mediu, 292
ieșire pentru copii tipărite, 293
IIS (Internet Information Server)
configurare, 385-386
instalare, 380
servere Web, Java, 380
imagini, tehnici de afișare pentru vedere slabă, 589-590
imbricare
elemente, 121
etichete, 36, 50-51
probleme, extensibilitatea XHTML, 486
IML, 580
implementare
date, 114-115
DTD-uri, 114-117
creare, 112
IMS (Educom Instructional Management Systems Project), 617
sit Web, 617
InDelv XML Client, browser cu suport XML, 680
sit Web, 680
independența de platformă (XML), 14
indicare
DTD extern, 84
DTD local, 85
indicatoare whatToShow (DOM Level 2), 213-214
indicator de apariții, 72-73
indicatori, whatToShow (DOM Level 2), 213-214
Information & Content Exchange (ICE), 59
sit Web, 618
inițiative (securitate/confidențialitate), 368-370
instalare
IIS, 380
JWSDK, 382-383
PWS, 380
XML4J, analizorul XML Java, 382
instanțierea claselor, 254
instrucțiuni Dim (obiecte COM), 419
instrucțiuni DOCTYPE (DTD-uri), 647
Instrument Markup Language, 580
instrumente
4DOM, 682
4XPath, 683
4XSLT, 683
conversie
David Raggett's HTML Tidy, 682
HTML-Kit from Chami.com, 682
Microstar FastTAG, 682
dezvoltare
Cocoon, 681
Multidoc Pro Publisher, 681
Near & Far Designer, 681
Python and XML Processing, 681
Visual XML, 681
XMetaL 1.2, 681
XML <PRO>, 681
XML Spy, 681
XML Testbed, 681
XMLwriter, 681
DTD-uri, depanare, 122-123
Extensibility XML Authority, 683
FOP, 683
IBM alphaWorks Tool Suite, 683-685
instrumente de criptare a fișierelor, 367
instrumente de securitate Windows, 377-378
Poet Content Management Suite, 683
Style One, 682
TopStyle, 273-274
Xalan, 683
XT, 682
instrumente de conversie
David Raggett's HTML Tidy, 682
HTML-kit from Chami.com, 682
Microstar FastTag, 682
instrumente de conversie XML Microstar FastTAG, 682
situl Web, 682
instrumente de criptare a fișierelor, 367
instrumente de dezvoltare Visual XML, 680-681
sit Web, 680
instrumente de dezvoltare XML Cocoon, 681
sit Web, 681
instrumente de securitate Windows, 377-378
instrumente Xalan, 683
sit Web, 683
instrumente XT, 682
sit Web, 682
int (întregi), grupul de lucru XML Schema al W3C, 159
integrare
depozite de date XML, 441
XML în ERP, 449-450
Interactive Financial Exchange (IFX), sit Web, 618
Interface Definition Language. *Vezi și* **IDL**
interfața Atribute List, 233
interfața AttributeList, 235
interfața Attributes 245-247
interfața ContentHandler, 233-236
interfața DocumentHandler, 233, 237
interfața DTDHandler, 233-237
interfața EntityResolver, 233, 239
interfața ErrorHandler, 233, 238
interfața Locator, 233
interfața LocatorImpl, 244
interfața Parser, 233, 245
interfața traversal (DOM), atribute/metode, 211
interfața XMLFilter, 233
interfața XMLReader, 233, 252
interfață utilizator (CSS), 737-739
sit Web, 621
interfețe
AttributeList, 233
AttributeList, 235
Attributes, 233-236
ContentHandler, 233-236
diferențe între DOM Level 1 și DOM Level 2, 209-212
DocumentHandler, 233-237
DTDHandler, 233-237
EntityResolver, 233-238
ErrorHandler, 233-238
Locator, 233
LocatorImpl, 244
orientate spre obiecte, operații, 425-426
Parser, 245
XMLFilter, 233
XMLReader, 233, 252
interfețe (DOM)
atribute/metode, 217
interfețe extinse, 219
interfețe fundamentale, 218-219
metode, 216-217
interfețe Cascading Style Sheets (CSS) (DOM), 210
interfețe eveniment (DOM), 211
interfețe fundamentale (DOM), 218-219
interfețe HTML (DOM), 210
interfețe nod (DOM)
atribute, 216-217
metode, 216-217
interfețe pentru foi de stil (DOM), 210
interfețe view (DOM), 210
interfețele range (DOM), 211
interMute, 369
Internal Revenue Service (IRS), 369
International Layout in CSS, sit Web, 621
International Standard Serial Numbers. *Vezi și* **ISSN**
International Standards Organization. *Vezi și* **ISO**
Internet Explorer
foi de stil, 272
limbi, coregrafia XML, 631
Internet Explorer 5 (Microsoft), browser cu suport XML, 680
Internet Explorer 5
downloads, 406
Internet Explorer, 631. *Vezi și* **Microsoft Internet Explorer**
Internet Information Server. *Vezi și* **IIS**

Internet Junkbuster Proxy, 369
Internet Protocol (IP), 351
Internet Worm, 352-353
Interoperable Object Reference. ***Vezi și*** **IOR**
interval de frecvențe, 299
intervale, selectare, 203
inversarea codării URL, 389-390
IP (Internet Protocol), 351
IRS (Internal Revenue Service), 369
ISBN-uri
declararea zonelor de nume, 139
zone de nume, situri Web, 139-140
ISO (International Standards Organization), 613
sit Web, SGML specificațiile, 613-614
ISO-12083
DTD-uri, private și publice, 613-614
standarde, sit Web, 614
XMLXperts, sit Web, 614
ISSN (International Standard Serial Numbers), 139
declararea zonelor de nume, 139
zone de nume, situri Web, 139-140
istoria XML, 26
conceptul Generic Coding, 28
convenții de marcare în ebraica veche, 27
convenții de marcare în latina veche, 27-28
GML (Generalized Markup Language), 28
inițiative recente, 28
Runoff, 28
SGML (Standard Generalized Markup Language), 28
iterație, construirea DTD-urilor, 112
încapsularea obiectelor, 424
începutul dorit al elementelor, determinare (SMIL), 516
începutul efectiv al elementelor, determinare (SMIL), 516
începutul implicit al elementelor, determinare (SMIL), 513-514
încorporarea elementelor, DTD-ul XHTML, 99-101
înfrumusețare. ***Vezi și*** **tipărire sugestivă, 490**
înglobare
conținut XML, etichete XHTML, 109
script-uri, 650
înregistrări, scheme XML, 163
întregi (int), grupul de lucru XML Schema al W3C, 159
învelișuri Perl, aplicații Java CGI, 383-384

J

James Clark XT, sit Web, 397
Java, 255-256
CGI
aplicații, învelișuri Perl, 383-384
execuție, 383
HTML, execuție, 386-387
programe, execuție, 404
implementarea SAX, 229
profiluri utilizator, P3P, 342
servere Web
IIS, 380
PWS, 380
serverul Web iPlanet, 381-382
serverul Web Java, 381
servere, 379-380
situri web Sun, 436
software, 379-380
Java Development Kit (JDK), 382
Java RMI. ***Vezi și*** **RMI**
Java SDK (Software Development Kit), 229
Java Speech Markup Language (JSML), 618
Java Virtual Machine. ***Vezi și*** **JVM**
Java Web Server Development Kit (JWSDK). ***Vezi și*** **JWSDK**
Java Web Server
oprire, 404
servere Web, Java, 381
JavaScript
șiruri comentariu / /, 145
tehnici de afișare pentru vedere slabă, 591-592
JCALS (Joint Computer-aided Acquisition and Logistics Support), 612
JDK (Java Development Kit), 382
JEDMICS (Joint Engineering Data Management Information and Control System), 612
Joint Computer-aided Acquisition and Logistics Support (JCALS), 612
Joint Engineering Data Management Information and Control System (JEDMICS), 612
JSDK (Java Servlet Development Kit), 382-383
JSML (Java Speech Markup Language), 618
sit Web, 618
JUMBO, browser cu suport XML, 680
sit Web, 680
Juniper Networks, 355
JVM (Java Virtual Machine), 436
RMI, 436
JWSDK (Java Web Server Development Kit), 381
descărcare, 382
instalare, 382
sit Web, 382

K

kiloHertz (kHz), 300
kit-uri de instrumente, legare, 183

L

legare, 183. ***Vezi și*** **XLink, XPointer**
comportamentul legăturii, 198
depanarea legăturilor întrerupte, 204
kit-uri de instrumente, 183
la documente numai pentru citire, 200
legături externe, 197
legături extinse, 186-189, 196
legături in-line, 196-197
legături simple, 186-187, 194-195
verificarea sintaxei legăturii, 204
XLink, 184-185
legături
SMIL (Synchronized Multimedia Integration Language), 520
elementul a, 520-521
elementul ancoră, 521
tehnici de afișare pentru vedere slabă, 590-591
verificare, 652
legături (HTML), 180
creare, 180-181
probleme, 181
legături externe, 197
legături externe, 284-285
legături in-line, 186, 196-197
legături întrerupte, depanare, 204
legături simple, 194-195
transcludere, 185
XLink, 186-187
legende, SMIL (Synchronized Multimedia Integration Language), 528-530
limbaje comune
ERP, 450-451
între sistemele de întreprindere, 452
limbaje de programare, suport DOM, 224-225
limbaje de stil. ***Vezi și*** **foi de stil**
limbaje pentru asigurarea datelor, 442
limbajul de machetare de bază (SMIL), 505
limbi, 460. ***Vezi și*** **vocabulare**
coregrafia XML
Microsoft Internet Explorer, 631
Mozilla, 629-631
Opera, 632
DTD, 59
ERP, obișnuit, 450-451
etichete (HTML), 485
grupul de lucru XML Schema al W3C, 161
multiple, coregrafie XML, 627-628
natural, coregrafie XML, 627
publicarea în limbi străini, 584-586
recunoaștere, 37
coregrafia XML, 629-631
stil.** ***Vezi și*** **foi de stil
limbi străine
DTD-uri, 59
publicare de documente în, 584-586
recunoaștere, 37
limbile Netscape Mozilla, 629. ***Vezi și*** **Mozilla**
linii noi, 51
lista poștală www-mobile@w3.org
liste (CSS), 730
listinguri
casetă de text mobilă (cod sursă XHTML), 295
cheia PGP a autorului, 365-366
CML, 576
DTD-ul XHTML 1.01 Tranzitoriu, 653-675
exemplu de schemă XML, 151-152
foaie de stil care simulează o casetă de text mobilă, 294
foi de stil în cascadă (CSS), 259-260, 269-272
foi de stil în cascadă Level 1 (CSS1), 260-261
foi de stil în cascadă Level 2 (CSS2), 262-263
gramatica XML în EBNF, 640-645
inversarea codării URL, 389-390
înregistrare XML a clienților, 153-155
scheme asociate, 155-156
învelișuri Perl pentru aplicații CGI Java, 383-384
limbajul extensibil al foilor de stil (XSL), 266-267
limbajul extensibil al foilor de stil, 274-277, 323-328
MathML, 569-575
ParseXML, 388
proiectul librăriei (exemplu de accesibilitate)
pagina de intrare XHTML pentru lista de librării, 603-604
pagina de navigare a sitului librăriei, 605-606
textul de prezentare al librăriei, 600-601
structura unui atribut XML Schema, 158
structura unui element XML Schema, 157, 158

un flacon de aspirină desfăcut pe componente, 22
XHTML și XML, 278-279
XML Schema, 307-308
XSL, 266-268, 274-277, 323-328
listinguri de cod. *Vezi și* listinguri
literale, 63
localizare, coregrafia XML, 628-629
localizarea DTD-urilor, 127-128
locator (XLink)
atribute (XLink), 189
elemente, 192
logică circulară, (DTD-uri), 122
long (întregi), grupul de lucru XML Schema al W3C, 159

M

machete
element (SMIL), 504-505
flexibilitate, 14
MAchine Readable Cataloging (MARC), 615, 615
Macsyma, 575
Manex (ERP), 455
sit Web, 454
Manufacturing Resource Planning (MRP), 454
Mapics (ERP), 454
sit Web, 454-455
Maple, 575
MARC (DTD-uri), 615
MARC (MAchine Readable Cataloging), 615, 615
marcaj condițional (DTD), 89-90
marcaje, 48
atribute, 54-56
entități, 49-50
etichete, 48-51
marcaje de citat (CSS), 731-732
Math Markup Language. *Vezi și* MathML
MathCad, 575
Mathematica, 540, 569-575
Mathematica, aplicație cu suport XML, 685
sit Web, 685
Mathematical Markup Language (MathML), 621
MathEQ, 540
MathML (Mathematical Markup Language), 59, 539-540, 621
algebră liniară, 564
algebră, 560-561
aritmetică, 560-561
browserul Amaya, 569
calcule, 561-562
combinarea matematicii cu textul, 568-569
conversii, 542
editorul de ecuații WebEQ, 575
elemente conținut. *Vezi și* elemente de sintaxă în acest spațiu
elemente de prezentare, 542-545
atribute, 543-545
combinarea cu elemente de sintaxă, 566-567
maction, 554
malignmark, 554
malignmark, 554
mchar, 556
menclose, 554
meqno, 554
merror, 549
mfenced, 543, 550-551
mfrac, 544, 548
mglyph, 555-556
mi, 545
mmultiscripts, 552
mn, 545
mo, 545-546
mover, 552
mpadded, 549-550
mphantom, 550
mprescripts, 552
mroot, 548
mrow, 548
ms, 547
mspace, 547
msqrt, 548
mstyle, 548-549
msub, 551
msubsup, 551-552
msup, 551
mtable, 553
mtd, 553-554
mtext, 546
mtr, 553
munder, 552
munderover, 552
sau elemente de sintaxă, 566-567
elemente de sintaxă, 556-557
abs, 561
and, 560
annotation, 564
annotation-xml, 565
apply, 558
approx, 565
arcos, 563
arcsin, 563
arctan, 563
arg, 565
bvar, 562
ci, 557
cn, 557
compose, 560
condition, 559
conjugate, 561
cos, 563
cosh, 563
cot, 563
coth, 563
csc, 563
csch, 563
csymbol, 565
curl, 565
declare, 559
degree, 562
determinant, 564
diff, 561
divergence, 565
divide, 560
eq, 561
equivalent, 565
exists, 561
exp, 560
factorial, 560
fn, 558
forall, 560
gcd, 560
geq, 561
grad, 565
gt, 561
ident, 560
imaginary, 565
implies, 560
in, 562
int, 561
intersect, 562
interval, 558
inverse, 558-559
lambda, 559
laplacian, 565
leq, 561
limit, 563
list, 562
ln, 561
log, 561
lowlimit, 562
lt, 561
matrix, 564
matrixrow, 564
max, 560
mean, 563
median, 564
min, 560
minus, 560
mode, 564
moment, 564
neq, 561
not, 560
notin, 562
notprsubset, 562
notsubset, 562
or, 560
outerproduct, 565
partialdiff, 561
plus, 560
power, 560
product, 563
prsubset, 562
quotient, 560
real, 565
reln, 558
rem, 560
root, 560
sau elemente de prezentare, 566-567
scalarproduct, 565
sdev, 563
sec, 563
sech, 563
selector, 564
semantics, 564
sep, 559
set, 562
setdiff, 562
sin, 563
sinh, 563
size, 565
subset, 562
sum, 562
tan, 563
tanh, 563
tendsto, 563
times, 560
transpose, 564
union, 562
uplimit, 562
variance, 563
vector, 564
vectorproduct, 565
xor, 560
elementul de interfață, 565-566
importanța, 540-542
limitări, 539-540
listinguri, 569-575
logică, 560-561
Macsyma, 575
Maple, 575
MathCAD, 575
Mathematica, 569-575
mulțimi, 562
recomandări, 539-540, 544
referințe, 539-540
relații, 561
reprezentare a semnificațiilor, 564-565
scheme de machetare, 547-551
expresii, 554
indici, 551-552

matrice, 552-554
tabele, 552-554
specificația MathML 2.0, 555-556, 564
statistică, 563-564
șiruri și serii, 562-563
transferarea de la un mediu la altul, 539-540
trigonometrie, 563
zonă de nume, 542
MathML, sit Web, 621
MathType, 540
MathType, aplicație cu suport XML, 685
sit Web, 685
matrice (MathML), 552-554
măsurarea de valori
frecvențe (auditive), 300
timp, 300
unghiuri, 300
măsurarea unghiurilor, 300
MEARS (Multi-user Engineering Change Proposal Automated Review System), 612
mecanisme de recuperare (HTML)
elemente imbricate incorect, 486
instrucțiuni de procesare, 486
spații albe, 486
medii, coregrafia XML
interactive, 632
multiple, 632
mediu care se modernizează (ERP), 452-453
mediu interactiv, coregrafia XML, 633
Megginson, David, 230
mesaje (GUI), 425
Meta Content Framework (MCF), sit Web, 618
metadate, 460-461
combinarea schemelor de metadate, 474-475
date, RDF, 473
Dublin Core, 460-473
PICS, 460-461, 470-471
Adequate.com, 472
RSACi, 471
Safe for Kids, 472
Safe Surf, 472
Vancouver Webpages Rating Service, 471-472
RDF, 460-463
colecții, 477-478
definirea vocabularelor RDF universale, 464-465
depanare, 479-480
funcționare, 462-463
proprietăți, 475-476
resurse, 475
sintaxă, 473
utilizări, 478-479
valori, 477
valorile Dublin Core, 473
vizualizare, 460-461
meta-limbaj, 17
metoda addAttribute(), 235
metoda AttributeListImpl(), 235
metoda Attributes(), 235
metoda characters(), 236
metoda clear(), 236
metoda createXMLReader(), 236
metoda declarePrefix(), 236
metoda DefaultHandler(), 236-237
metoda doGet(), trimiterea unui formular, 393
metoda endDocument(), 237
metoda endElement(), 237
metoda endPrefix(), 238
metoda endPrefixMapping(), 238
metoda error(), 238
Metoda fatalError(), 238
metoda getByteStream(), 238
metoda getCharacterStream(), 238
metoda getColumnNumber(), 238
metoda getDeclaredPrefixes(), 239
metoda getDTDHandler(), 239
metoda getEncoding(), 239
metoda getEntityResolver(), 239
metoda getErrorHandler(), 239
metoda getException(), 239
metoda getFeature(), 239
metoda getIndex(), 239-240
metoda getLength(), 240
metoda getLineNumber(), 240
metoda getLocalName(), 240
metoda getMessage(), 240
metoda getName(), 240
metoda getParent(), 240
metoda getPrefix(), 241
metoda getPrefixes(), 241
metoda getProperty(), 241
metoda getPublicId(), 241
metoda getQName(), 241
metoda getSystemId(), 241
metoda getType(), 241-242
metoda getURI(), 242
metoda getValue(), 242-243
metoda HandlerBase(), 243
metoda ignorableWhitespace(), 243
metoda init(), 399-400
metoda InputSource(), 243-244
metoda ITERATOR (DOM), 212-213
filtre, 214-215
indicatorii what ToShow, 213-214
metoda LocatorImpl(), 244
metoda makeParser(), 244
metoda NamespaceSupport(), 244
metoda notationDecl(), 244-245
metoda parse(), 245
metoda ParseAdapter(), 245
metoda popContext(), 245
metoda processingInstruction(), 245-246
metoda processName(), 246
metoda pushContext(), 246
metoda removeAttributte(), 246
metoda reset(), 246
metoda resolveEntity(), 246
metoda SAXException(), 246-247
metoda SAXNotRecognizedException(), 247
metoda SAXNotSupportedException(), 247
metoda SAXParseException(), 247
metoda setAttribute(), 247
metoda setAttributeList(), 247
metoda setAttributes(), 247
metoda setByteStream(), 247
metoda setCharacterStream(), 247
metoda setColumnNumber(), 247
metoda setContentHandler(), 247-248
metoda setDocumentHandler(), 248
metoda setDocumentLocator(), 248
metoda setDTDHandler(), 248-249
metoda setEncoding(), 249
metoda setEntityResolver(), 249
metoda setErrorHandler(), 249
metoda setFeature(), 249
metoda setLineNumber(), 249
metoda setLocale(), 249
metoda setLocalName(), 250
metoda setParent(), 250
metoda setProperty(), 250
metoda setPublicId(), 250
metoda setQName(), 250
metoda setSystemId(), 250
metoda setType(), 250
metoda setURI(), 250
metoda setValue(), 250
metoda skippedEntity(), 250
metoda startDocument(), 250-251
metoda startElement(), 251
metoda startPrefixMapping(), 251
metoda toString(), 252
metoda TreeWalker (DOM), 212-213
filtre, 214-215
indicatoare whatToShow, 213-214
metoda unparsedEntityDecl(), 252
metoda warning(), 252
metoda XMLFilterImpl(), 252
metoda XMLReaderAdapter(), 252-253
metode
addAttribute(), 235
AttributeListImpl(), 235
AttributesImpl(), 235-236
characters(), 236
clear(), 236
createXMLReader(), 236
declarePrefix(), 236
DefaultHandler(), 236-237
diferențe între DOM Level 1 și DOM Level 2, 209-210
Document Object Model, 206-207
doGet(), 393
endDocument(), 237
endElement(), 237
endPrefixMapping(), 238
error(), 238
fatalError(), 238
filtre, 214-215
getByteStream(), 238
getCharacterStream(), 238
getColumnNumber(), 238
getDeclaredPrefixes(), 239
getDTDHandler(), 239
getEncoding(), 239
getEntityResolver(), 239
getErrorHandler(), 239
getException(), 239
getFeature(), 239
getIndex(), 239-240
getLength(), 240
getLineNumber(), 240
getLocalName(), 240
getMessage(), 240
getName(), 240
getParent(), 240
getPrefix(), 241
getPrefixes(), 241
getProperty(), 241
getPublicId(), 241
getQName(), 241
getSystemId(), 241
getType(), 241-242
getURI(), 242
getValue(), 242-243
HandlerBase(), 243

ignorableWhitespace(), 243
indicatoare whatToShow, 213-214
init(), 399-400
InputSource(), 243-244
interfața Traversal (DOM), 217
Iterator (DOM), 212-213
LocatorImpl(), 244
makeParser(), 244
metode ale interfeței nod (DOM), 216-217
metode DOM, 220-223
moștenire CSS, 221
moștenire principală, 220-221
moștenirea evenimentelor, 221
moștenirea HTML, 221-223
NamespaceSupport(), 244
notationDecl(), 244-245
obiecte, 424-425
    încapsulare, 424
parse(), 245
ParserAdapter(), 245
popContext(), 245
processingInstruction(), 245-246
processName(), 246
pushContext(), 246
removeAttribute(), 246
reset(), 246
resolveEntity(), 246
SAXException(), 246-247
SAXNotRecognizedException(), 247
SAXNotSupportedException(), 247
SAXParseException(), 247
setAttribute(), 247
setAttributeList(), 247
setAttributes(), 247
setByteStream(), 247
setCharacterStream(), 247
setColumnNumber(), 247
setContentHandler(), 247-248
setDocumentHandler(), 248
setDocumentLocator(), 248
setDTDHandler(), 248-249
setEncoding(), 249
setEntityResolver(), 249
setErrorHandler(), 249
setFeature(), 249
setLineNumber(), 249
setLocale(), 249
setLocalName(), 250
setParent(), 250
setProperty(), 250
setPublicId(), 250
setQName(), 250
setSystemId(), 250
setType(), 250
setURI(), 250
skippedEntity(), 250
startDocument(), 250-251
startElement(), 251
startPrefixMapping(), 251
toString(), 252
TreeWalker (DOM), 212-213
unparsedEntityDecl(), 252
warning(), 252
XMLFilterImpl(), 252
XMLReaderAdapter(), 252-253
Microsoft
(DOM), 208
CSS, 285-286
Equation Editor, 539
standarde deschise, 406-407
Windows DNA, 407-408
XSL, 322-323
Microsoft Chrome, coregrafia XML, 634. ***Vezi și*** **Chrome**
Microsoft DHTML (Microsoft Dynamic HTML), 633. ***Vezi și*** **DHTML**
comportamente, coregrafia XML, 631
Microsoft Intenet Explorer, 631. ***Vezi și*** **Internet Explorer**
foi de stil, 272
limbi, coregrafia XML, 631
situl Web, 624
Microsoft Transaction Server (MTS), 434
migrare (ERP)
rezultate, 454
strategii, 453, 456
Milbrandt, Dr. Eric, 372
milisecunde (ms), 300
mini aplicații (applet-uri), scriere, 392
minimizare
HTML, 481-482
Pretty HTML, 483
programul Tidy, 482-483
translatarea în XHTML, 482-483
Missile Industry Supply-chain Transaction Infrastructures (MISTI), 59
MISTI (Missile Industry Supply-chain Transaction Infrastructures), 59
modele de conținut, 71
scheme XML, definire, 162
modele de obiecte
COM
DCOM, 431-432
dezavantaje, 431
MTS, 434
ORPC, 432
modele, conținut, 71
modelul de obiecte COM+, 431
modificare
denumiri de bloc, 116
DTD-uri, 109-110
entități parametru, 95-97
forță brută, 104
modificare, DTD, 90-91
modificarea DTD-urilor, 104
modificări, scheme XML
analiza, 146
rezistența la, 145-146
MOLMOL, 576
month, grupul de lucru XML Schema al W3C, 160
Morris, Robert, 352
Mortice Kern Systems, 377
moștenire, 254-255
foi de stil în cascadă, 712
moștenire CSS (DOM), 221
moștenire evenimente (DOM), 221
moștenire HTML (DOM), 221-223
moștenire principală (DOM), 220-221
moștenirea CSS (DOM), 221
moștenirea evenimentelor (DOM), 221
moștenirea HTML (DOM), 221-223
moștenirea principală (DOM), 220-221
Mozilla, 629. ***Vezi și*** **Netscape Mozilla**
browser
foi de stil, 264
limbi, coregrafia XML, 629-631
sit Web, 264, 624
suport CSS, 286
Mozilla, browser cu suport XML, 680
MRP (Manufacture Resource Planning), 454
MSXML (analizor XML), 406
downloads, 406
MTS (Microsoft Transaction Server), 434
Muffin, 369
Mulberry Technologies, 282
MultiDoc Pro browser, 279
MultiDoc Pro Publisher, instrumente de dezvoltare XML, 681
sit Web, 681
multimedia, tehnici pentru vedere slabă, 593
multiple
alfabete, coregrafia XML, 632
coregrafia XML
fluxuri de date, 634-635
utilizatori, 635
DTD-uri cu zonă de nume, 129-130
medii, coregrafia XML, 632
Multi-user Engineering Change Proposal Automated Review System (MEARS), 612

N

Namespaces in XML, sit Web, 621
NANP (North American Numbering Plan), 152
narațiuni (SMIL), 528-530
National Stock Numbers (NSN), 612
ncname, grupul de lucru XML Schema al W3C, 162
Near & Far Designer, instrumente de dezvoltare XML, 681
sit Web, 681
negativeInteger, grupul de lucru XML Schema al W3C, 160
Netopia, 355
Netscape
Document Object Model (DOM), 208
foi de stil, 272
suport CSS, 303
Netscape Cascading Action Systems, 633. ***Vezi și*** **Cascading Action Systems**
coregrafia XML, 633
Newspaper Association (NAA), sit Web, 618
NMTOKEN, 76
atribute, 79
nmtoken, grupul de lucru XML Schema al W3C, 161
nmtokens, grupul de lucru XML Schema al W3C, 161
noduri (XPath), 171-172
noduri atribut, 171
noduri context, 172
noduri text, 171-172
relații între, 172
atribut, 173
copil, 172
descendent sau el însuși, 174
descendent, 172
el însuși, 173
fratele precedent, 173
părinte, 172
precedentul, 173
strămoș sau el însuși, 174
strămoș, 173
următorul frate, 173
următorul, 173
zonă de nume, 173
noduri atribut (XPath), 171
noduri context (XPath), 172
noduri text (XPath), 171-172
nonNegativeInteger, grupul de lucru XML Schema al W3C, 160

nonPositiveInteger, grupul de lucru XML Schema al W3C, 160
normalizare, 51-52
North American Numbering Plan (NANP), 152
Norton Internet Security 2000 al Symantec, 359
Norton Internet Security 2000, 359
not invented here (NIH - nu a fost inventat aici), 106
notation, grupul de lucru XML Schema al W3C, 161
notația cu acolade, Opera, 296
notația cu punct, 286
notații, 58, 65-66, 75-76
atribut NOTATIONS, 648-649
declarare, 64-65
DTD, 63
grupul de lucru XML Schema al W3C, 161
sintaxă, 62-63
NPT (Normal Playing Time) valori, 512
NSN (National Stock Numbers), 612
nume
bloc, modificare, 116
elemente, adunare, 114-115
entități, DTD-uri blindate, 124
grupul de lucru XML Schema al W3C, 161
nume bloc, modificare, 116
nume calificat (qname), 161
nume calificate, 133-134
etichete XHTML, 134-136
nume de elemente arbitrare, 286
numerotare
CSS, 732
XSL, 322

O

OASIS (Organization for the Advancement of Structured Information Standards), 615
sit Web, 59, 615
obiect Request (ASP), 409
obiecte
constructori, 253-254
definite, 253
destructori, 253-254
distribuite, 423-424
GUI, 425
IDL, 424-425
metode, 424
încapsulare, 424
obiecte ASP
obiectul Request, 409
obiectul Response, 409-411
obiecte COM, 415-421
Active Server Pages (ASP), 415-421
creare în Visual Basic, 416-420
ORB, 423-424
verificatoare ortografice, 423-424
obiecte COM, 415-421
Active Server Pages (ASP), 415-416, 420-421
creare cu Visual Basic, 416-420
arbori de ieșire în XML, creare, 418-419
condiții de eroare, verificare, 418
DLL-uri, creare, 416-417
fișier DLL, creare, 420
prelucrare XML, 418
obiecte distribuite, 423-424
obiectul Response (ASP), 409-411
Object Remote Procedure Call (ORPC), 432
Object Request Brokers. *Vezi și* ORB-uri
obținerea clasei XmlDocument, 397
OIF (Open Integration Framework), 456
Online Computer Library Center (OCLC), 465
Ontology Markup Language, sit Web, 618
OOP (programare orientată spre obiecte), 253-254
clase, 254
Java, 255-256
moștenire, 254-255
obiecte
constructori, 253-254
definire, 253
destructori, 253-254
Python, 255-256
Open Content Syndication (OCS), sit Web, 618
Open Integration Framework (OIF), 456
Open Profiling Standard (OPS), 341, 368-369
Open Software Description Format (OSD), sit Web, 618
Open Trading Protocol (OTP), sit Web, 618
OpenMLS, Real Estate DTD Design, sit Web, 618
OpenTag, sit Web, 618
Opera
browser, foi de stil, 262, 273, 323
limbi, coregrafia XML, 632
notație cu acolade, 296
sit Web (profiluri utilizator), 334-335
suport CSS, 286
oprirea Java Web Server, 404
OPS (Open Profiling Standard), 341, 368-369
OPS/P3P, profiluri utilizator, 341
Oracle (ERP), 454-455
sit Web, 454-455
ORB-uri (Object Request Brokers), 423-428
COM
cerere, 426-427
combinarea cu componentele Microsoft, 431
comunicare între, 427-428
CORBA, 429
cerere, 426-427
citirea resurselor, 429
depanare, 429
DII, 429
IDL, 429
ORB-uri, 429
GUI, mesaje, 425
Internet, 423-424
IOR, 403
obiecte, 423-424
protocoale, comunicare, 425
RMI, cerere, 426-427
XMLTP, cerere, 426-427
ORB-uri Internet, 423-424
ORB-uri IOR (Interoperable Object Reference), 403
ordinea etichetelor, 21-23
Organization for the Advancement of Structured Information Standards (OASIS), 59, 615
orientare spre obiecte, 206-207
descriere, 15
DOM, 218-223
interfețe extinse, 219
interfețe fundamentale, 218-219
metode, 220-223
moștenire CSS, 221
moștenire de evenimente, 221
moștenire HTML, 221-223
moștenire principală, 220-221
interfețe, operații, 426
mecanismul de apel la distanță al procedurilor (RMI), 436
programare. *Vezi și* OOP
sisteme, 425
vedere (Document Object Model), 206-207
originile XML, 26
conceptul Generic Coding, 28
convenții de marcare în ebraica veche, 27
convenții de marcare în latina veche, 27-28
GML (Generalized Markup Language), 28
inițiative recente, 28-29
Runoff, 28
SGML (Standard Generalized Markup Language), 28-29
ORPC (Object Remote Procedure Call), 432
modelul de obiecte COM, 432

P

P3P (Platform for Privacy Preferences), 376, 621
profiluri utilizator
implementate cu Java, 342
OPS, 341
politici de examinare, 340
preferințe de confidențialitate, 337-338
schimb de politici de confidențialitate, 339-340
secvența evenimentelor, 338-339
sit Web, 621
standarde, situri Web, 340
pachete, 351
pachete
org.xml.sax, 232. *Vezi și* SAX (Simple API for XML)
clase, 234
excepții, 234
interfețe, 233
org.xml.sax.helpers, 234. *Vezi și* SAX (Simple API for XML)
pachetul org.xml.sax, 232. *Vezi și* SAX (Simple API for XML)
clase, 234
excepții, 234
interfețe, 233
pachetul org.xml.sax.helpers, 234. *Vezi și* SAX (Simple API for XML)
Paged Media Properties for CSS3, sit Web, 621
pagere, afișaje, 301
pagini (CSS), 732-734
Palm Pilots, 301
Panorama SGML Viewer, browser cu suport XML, 680
sit Web, 680
parametri, cod CGI
extragere, 388-389
scriere, 390-391
parametri de extragere, cod CGI, 388-389
paranteze, liste de elemente, 116
parole unice, creare, 362-363
parole/ID-uri
alegere, 362-363
profiluri utilizator, 336-337
adrese de e-mail, 336-337
particularizare
SMIL (Synchronized Multimedia Integration Language), 536-537
XHTML, 490
PeopleSoft (ERP), 456
perechi proprietate/valoare (CSS), 290

Perl XML, sit Web, 624
Persistent URL (PURL), 128
Personal Web Server (PWS), downloads, 406
PICS (Platform for Internet Content Selection), 460-461, 470-471
Adequate.com, 472
RSACi, 471
Safe for Kids, 472
Safe Surf, 472
Vancouver Webpages Rating Service, 471-472
PICS Signed Labels, sit Web, 621
PivotPoint (ERP), 454
sit Web, 455
Platform for Internet Content Selection (PICS), sit Web, 618
Platform for Privacy Preferences (P3P), 621
plug-in-uri, tehnici de afișare pentru vedere slabă, 593
Plume, browser cu suport XML, 681
sit Web, 681
plus (+) (EBNF), 646
Poet Content Management Suite, instrumente, 683
sit Web, 683
Poet, gestiunea resurselor de date, 443-444
pointeri
pointeri TEI, 198
XPointer-i, 198
adresare absolută, 201
adresare relativă, 200-201
depanare, 204
extensii, 202-203
funcții booleene, 203
funcții de localizare absolută, 203
legături la documente numai pentru citire, 200
selectare prin nume, 201-202
sintaxă, 198-199
porturi
porturi binecunoscute, 356-358
porturi dinamice, 358
porturi înregistrate, 358
porturi binecunoscute, 356-358
porturi dinamice, 358
porturi înregistrate, 358
positiveInteger, grupul de lucru XML Schema al W3C, 160
POST, apel CGI, 386-387
precizie (XML), 14
predicate (XPath), 179
prelucrare
comandă, 650
formulare, 394-395
instrucțiuni, 82, 284-285
HTML, mecanisme de recuperare, 486
preprocesare, extensibilitatea XHTML, 487-488
prezentări fotografice, RealSlideshow, 503
SMIL (Synchronized Multimedia Integration Language), 521-522
prezentări multimedia, SMIL, 522
PrivaSeek, sit Web (profiluri utilizator), 333
privire generală
ERP, 448-450
scheme XML, 143
procesarea tranzacțiilor, 433-434
procesoare
CDATA, 52
entități neanalizate, 56-58
linii noi, 51
spații albe, 51-52
Process Interchange Format XML (PIF-XML), sit Web, 618
producții (EBNF), 639-645
produse firewall existente, 359-361
profiluri
utilizator, 332
asistenți de confidențialitate, 343-345
BBBOnline, sit Web, 342
brokeri de confidențialitate, 336
declarații de confidențialitate, sit Web, 343
ID-uri/parole utilizator, 336-337
Microsoft LinkExchange, sit Web, 343
negociere, 333-334
Opera, sit Web, 335
OPS/P3P, 341
P3P Standards, sit Web, 340
P3P, 337-338
PrivaSeek, sit Web, 333
TRUSTe, sit Web, 340
TV Guide, sit Web, 333
utilizarea bazelor de date server-side, 335-336
utilizarea cookie-urilor, 334-335
programe CGI (Java), execuție, 386-387, 404
programul de analiză moleculară și afișare a moleculelor, 576
programul HTML-Kit, 37
programul Pretty HTML
minimizările HTML, 483-484
situri Web, 483
programul Tidy al lui Dave Raggett. ***Vezi și*** **programul Tidy**
programul Tidy, 37, 482-483
minimizarea HTML, 482-483
situri Web, 482
proiect pentru o librărie feministă (exemplu de accesibilitate)
aspectul paginii, 602-603, 607-608
cadre, 601-602
pagina de intrare XHTML a librăriei, 603-604
pagina de navigare, 605-606
textul pentru indexul librăriei, 600-601
proiectare
DTD-uri
adunarea datelor, 113-114
alcătuirea DTD-urilor, 112
colectarea cunoștințelor din domeniu, 113-114
principii, 120-121
scopuri, 114
situri Web (CSS), 283-284
proiectul librăriei (exemplu pentru accesibilitate)
aspectul paginii, 602-603, 607-608
cadre, 601-602
pagina de intrare XHTML a librăriei, 603-604
pagina de navigare, 605-606
textul pentru indexul librăriei, 600-601
Project Gutenberg, sit Web, 128
prologuri (DTD-uri), 646
proprietăți (CSS), 287
proprietăți de comportament, 302
propuneri, scheme XML, 166
protocoale
fără conexiune, 351
IP (Internet Protocol), 351
ORB-uri, comunicare, 425
TCP (Transmission Control Protocol), 352
TCP/IP, 351
criptare, 368
IP (Internet Protocol), 351
pachete, 351
TCP (Transmission Control Protocol), 352
XML (SOAP), 440-441
protocoale bazate pe XML (SOAP), 440-441
protocoale fără conexiune, 351
pseudo-clasa after, 290
pseudo-clasa before, 289
pseudo-clasa first-child, 289
pseudo-clasa first-letter, 289
pseudo-clasa first-line, 289
pseudo-clasa focus, 289
pseudo-clasa hover, 289
pseudo-clasa lang (cod de limbă), 289
pseudo-clasa link, 289
pseudo-clasa visited, 289
pseudo-clasă activă, 289
pseudo-clase, 289
Cascading Style Sheets, 714-715
pseudo-element activ, 289
pseudo-elemente, 289
Cascading Style Sheets, 714-715
CSS, 287
pseudo-elementul first-letter, 289
pseudo-elementul first-line, 289
pseudo-elementul link, 289
pseudo-elementul visited, 289
Psion, 301
puncte forte, scheme XML, 146-147
PURL (Persistent URL), sit Web, 128
zone de nume, 141-142
PWS (Personal Web Server)
configurare, 385-386
downloads, 406
instalare, 380
servere Web (Java), 380
sit Web, 380
Python and XML Processing, instrumente de dezvoltare XML, 681
sit Web, 681
Python, 255-256
implementarea SAX, 229
sit Web, 624

Q

qname (nume calificat), 161
grupul de lucru XML Schema al W3C, 161
Quicktime, 504

R

radiani (rad), 300
rădăcini, elemente, 115-117
RDF (Resource Description Framework), 166, 460-463
colecții, 477-478
definirea vocabularelor RDF universale, 464-465
depanare, 479-480
funcționare, 462-463
proprietăți, 475-476
resurse, 475
scheme XML, 167
sintaxă, 474

utilizare, 478-479
valori Dublin Core, 473
valori, 477
RDF Model and Syntax, sit Web, 621
RDF Schema, sit Web, 621
Real Estate Markup Language (RELML), 59
Real Estate Transaction Standard (RETS), 59
RealSlideshow de la RealNetworks, 503
RealSlideshow, 503
Recomandarea XML 1.0 adnotată de Bray, 31-32
Recomandarea XML 1.0
entități, 647-648
sit Web, 646
recurringDate, grupul de lucru XML Schema al W3C, 160
recurringDay, grupul de lucru XML Schema al W3C, 160
recurringDuration, grupul de lucru XML Schema al W3C, 160
recursivitate, 36
redare auditivă
editarea vocilor, 599
foi de stil, 596-599
medii țintă multiple, 599
redarea tabelelor, 300
redare, DTD-uri XHTML, 109
redefinirea ATTLIST, 94
referențiere
entități, 67
seturi de caractere, 687-688
Referință tehnică (XML), 639
referințe pentru foi de stil (XML), 396
reguli, foi de stil, 287
relația atribut (XPath), 173
relația copil (noduri XPath), 172
relația descendent (noduri XPath), 172
relația descendent sau el însuși (noduri XPath), 174
relația el însuși (noduri XPath), 173
relația fratele precedent (noduri XPath), 173
relația părinte (noduri XPath), 172
relația precedentul (noduri XPath), 173
relația strămoș (noduri XPath), 173
relația strămoș sau el însuși (noduri XPath), 174
relația următorul (noduri XPath), 173
relația următorul frate (noduri XPath), 173
relația zonă de nume (noduri XPath), 173
relații între noduri (XPath), 172
atribut, 173
copil, 172
descendent sau el însuși, 174
descendent, 172
el însuși, 173
fratele precedent, 173
părinte, 172
precedentul, 173
strămoș sau el însuși, 174
strămoș, 173
următorul frate, 173
următorul, 173
zonă de nume, 173
RELML (Real Estate Markup Language), 59
Remote Method Invocation. *Vezi și* RMI
Resource Data Framework, 460
Resource Description Framework. *Vezi și* RDF
restaurarea criptării puternice, 367
restaurarea criptărilor puternice, 367
RETS (Real Estate Transaction Standard), 59
rețele
securitate. *Vezi și* securitate VPN (Virtual Private Networks), 360-361
rezultate (ERP), 454
RFC-2112, sit Web, 621
Rice, Stanley, 28
Rich Site Summary (RSS), sit Web, 618
Rich Text Format. *Vezi și* RTF
RMI (Remote Method Invocation), 436
clase interfață, definire, 437-438
JVM. 460
mecanism de apel la distanță al procedurilor orientat spre obiecte, 436
Roman Czyborra, sit Web, 45
RosettaNet, 59
RosettaNet, sit Web, 619
RSACi, 471
RTF (Rich Text Format), 442
stocarea datelor, 442
Ruby, sit Web, 621
Runoff, 28
rutere, 349-350
avantaje, 354-355
cheltuieli, 354
stiva protocolului TCP/IP, 350-352
IP (Internet Protocol), 351
pachete, 351
TCP (Transmission Control Protocol), 352

S

Sable, Speech Synthesis Markup Language, sit Web, 619
SAE J2008, 59
SAE J2008, Automotive Industry Standard, sit Web, 619
Safe for Kids, 472
Safe Passage, 367
Safe Surf, 472
Saltzer, Jerry, 28
SAP R/3 (ERP), 454-455
SAP, sit Web, 454
SAX, 238 (Simple API for XML)
analizoare, 229
avantaje, 226-228
clase
AttributeListImpl, 235
AttributesImpl, 235-236
DefaultHandler, 236-237
HandlerBase, 243
InputSource, 243-244
LocatorImpl, 244
NamespaceSupport, 244
ParserAdapter, 245
ParserFactory, 245
XMLReaderAdapter, 252-253
XMLReaderFactory, 253
depanare, 256
excepții
SAXException, 246-247
SAXNotRecognizedException, 247
SAXNotSupportedException, 247
SAXParseException, 247
interfețe
AttributeList, 235
Attributes, 235
ContentHandler, 236
DocumentHandler, 237
DTDHandler, 237
EntityResolver, 238
ErrorHandler, 238
Locator, 244
Parser, 245
XMLReader, 252
metode
addAttribute(), 235
AttributeListImpl(), 235
AttributesImpl(), 235-236
characters(), 236
clear(), 236
createXMLReader(), 236
declarePrefix(), 236
DefaultHandler(), 236-237
endDocument(), 237
endElement(), 237
endPrefixMapping(), 238
error(), 238
fatalError(), 238
getByteStream(), 238
getCharacterStream(), 238
getColumnNumber(), 238
getDeclaredPrefixes(), 239
getDTDHandler(), 239
getEncoding(), 239
getEntityResolver(), 239
getErrorHandler(), 239
getException(), 239
getFeature(), 239
getIndex(), 239-240
getLength(), 240
getLineNumber(), 240
getLocalName(), 240
getMessage(), 240
getName(), 240
getParent(), 240
getPrefix(), 241
getPrefixes(), 241
getProperty(), 241
getPublicId(), 241
getQName(), 241
getSystemId(), 241
getType(), 241-242
getURI(), 242
getValue(), 242-243
HandlerBase(), 243
ignorableWhitespace(), 243
InputSource(), 243-244
LocatorImpl(), 244
makeParser(), 244
NamespaceSupport(), 244
notationDecl(), 244-245
parse(), 245
ParserAdapter(), 245
popContext(), 245
processingInstruction(), 245-246
processName(), 246
pushContext(), 246
removeAttribute(), 246
reset(), 246
resolveEntity(), 246
SAXException(), 246-247
SAXNotRecognizedException(), 247
SAXNotSupportedException(), 247
SAXParseException(), 247
setAttribute(), 247
setAttributeList(), 247
setAttributes(), 247
setByteStream(), 247
setCharacterStream(), 247
setColumnNumber(), 247
setContentHandler(), 247-248
setDocumentHandler(), 248
setDocumentLocator(), 248
setDTDHandler(), 248-249
setEncoding(), 249
setEntityResolver(), 249
setErrorHandler(), 249

BIBL. CENTR. UNIV. „M. EMINESCU" IAȘI INFORMATICĂ

setFeature(), 249
setLineNumber(), 249
setLocale(), 249
setLocalName(), 250
setParent(), 250
setProperty(), 250
setPublicId(), 250
setQName(), 250
setSystemId(), 250
setType(), 250
setURI(), 250
skippedEntity(), 250
startDocument(), 250-251
startElement(), 251
startPrefixMapping(), 251
toString(), 252
unparsedEntityDecl(), 252
warning(), 252
XMLFilterImpl(), 252
XMLReaderAdapter(), 252-253
pachetul org.xml.sax
clase, 234
excepții, 234
interfețe, 233
pachetul org.xml.sax.helpers, 234
tratarea evenimentelor, 230-232
versiuni, 226, 229-230
scalabilitate, probleme CGI, 391-392
Scalable Vector Graphics (SVG), 622
Schema for Object-Oriented XML. ***Vezi și*** **SOX**
Schema.net, sit Web, 619
scheme
argumente, noi limbaje pentru scheme, 143-145
Datatypes (tipuri de date), 148
XML-Data, 166
XSchema, 167
grupul de lucru XML Schema al W3C
binary, 161
boolean, 159
byte, 160
century, 161
date, 160
decimals, 159
double, 159
entities, 161
entity, 161
float, 159
id, 161
idref, 161
idrefs, 161
int (întregi), 159
integer, 159
language, 161
long (întregi), 159
month, 160
name, 161
ncname, 162
negativeInteger, 160
nmtoken, 161
nmtokens, 161
nonNegativeInteger, 160
nonPositiveInteger, 160
notation, 161
notations, 161
positiveInteger, 160
qname, 161
recurringDate, 160
recurringDay, 160
recurringDuration, 160
short (întregi), 160
string, 159
time, 160
timeDuration, 160
timeInstant, 160
timePeriod, 160
unsignedByte, 160
unsignedInt, 160
unsignedLong, 160
unsignedShort, 160
uri-reference, 161
year, 160
Primer, XML Schema Part 1, 148
Structures, XML Schema Part 2, 148
XML
analiză, 146
atribute locale, 152-153, 155-156
câmpuri de intrare cu format liber, 165
conlucrarea între aptitudini, 148
constrângerea conținutului elementului și atributului, 162-163
definire, 152
definirea atributelor, 158
definirea elementelor, 157
definirea modelelor de conținut, 162
definirea tipurilor de date, 158-161
extinderea tipurilor de date, 162
identificarea compromisurilor, 147-148
limitările verificării erorilor și a validității, 164-165
modificări în, 145-146
privire generală, 143
propuneri anterioare, 166
punctele forte, 146-147
RDF, 166
setarea valorilor prestabilite, 167
SOX, 167
utilizarea zonelor de nume, 148-151
validarea câmpurilor definite prin scheme, 150
validarea conținutului elementului, 163
verificarea caracterului complet al înregistrărilor, 163
XML Schema Part 0
XML Schema, listinguri, 307-308

scriere
applet-uri, 392
cod CGI
extragerea parametrilor, 388-389
parametri, 390-391
tipul de conținut, 388
handler-e de evenimente, SAX (API simplu pentru XML), 232
servlet-uri, 392
script-uri și foi de stil (HTML), 485
script-uri, inserare, 650
secțiuni (CDATA), 651
secunde (s), 300
Secure Sockets Layer (SSL), 374-375
securitate
criptare, 354, 363-364
asimetrică, 364-367
criptare puternică, restaurare, 367
criptare TCP/IP, 368
instrumente de criptare a fișierelor, 367
simetrică, 364
depanare
exemple, 376-377
instrumente de sistem Windows, 377-378
firewall-uri
definire, 355
HTTP (Hypertext Transfer Protocol), 358-359
porturi binecunoscute, 356-358
porturi dinamice, 358
porturi înregistrate, 358
produse firewall existente, 359-361
servere proxy, 356
funcții hash, 375-376
hackeri, 354
inițiative de securitate și confidențialitate, 368-370
Internet Worm, 352-353
P3P, 376
parole, alegere, 362-363
rutere, 349-350
avantaje, 354-355
cheltuieli, 354
stiva protocolului TCP/IP, 350-352
semnături digitale, 370
SMDL (Signed Markup Description Language), 371-374
XML semnat, 370-371
spoofing, 353-354
SSL (Secure Sockets Layer), 374-375
vulnerabilitatea datelor, 347-349
secvențe audio descriptive, SMIL (Synchronized Multimedia Integration Language), 531-534
secvențe cu salturi, 199
secvențe escape (CDATA), 82
selectarea după nume (XPointers), 201-202
selectarea nodurilor XPath, 175-177
selectori, 283-284
adăugarea valorilor proprietate la elemente, 288
Cascading Style Sheets, 712-713
clasă, 290-291
comenzi de foaie de stil, creare, 288
creare, 287
CSS, 286-287
elemente ca, 286
interval, 288
selectarea elementelor paragraph dintr-un element părinte chapter într-un document XML, 288
selectorul de clasă, 290
selectorul DIV.warning, 287
selectorul E + F, 287
selectorul E F, 287
selectorul E#myid, 287
selectorul E, 287
selectorul E:active, 287
selectorul E:first-child, 287
selectorul E:focus, 287
selectorul E:hover, 287
selectorul E:link, 287
selectorul E:visited, 287
selectorul E[], 302
semn de întrebare (?) (EBNF), 645-646
semnături digitale, 370
SMDL (Signed Markup Description Language), 371-374
XML semnat, 370-371
sensibilitate la tipul literelor
cuvinte cheie EBNF, 646
DTD-uri XHTML, 675
HTML, 484-485
numele tipurilor de medii, 292
XML, 36
servere
Java, 379-380
pentru mai multe platforme, 379
servere proxy, 356
transformarea XML în HTML, 412-415
Web
IIS, 380
iPlanet Web Server, 381-382
Java Web Server, 381
PWS, 380
servere pentru mai multe platforme, 379

servere proxy, 356
servere Web, Java
IIS, 380
iPlanet Web Server, 381-382
Java Web Server, 381
PWS, 380
serverul Web iPlanet, 381-382
Java, 381-382
servicii de validare, 710, 748
servlet-uri
calea de includere a claselor, 404
dezvoltare, 392
execuție, 404
formulare, prelucrare, 394-395
scriere, 391-392
structură, 392-393
tipul de conținut, 395
XML, cerere, 399-402
seturi de caractere
referire, 687-688
XHTML
entități caracter (sortate alfabetic), 698-703
entități caracter (sortate numeric), 703-708
entități ISO caractere speciale, 692-693
entități ISO Latin-1, 688-692
entități ISO matematice, grecești și caractere simbolice, 693-698
sfârșitul dorit al elementelor, determinare (SMIL), 515
sfârșitul efectiv al elementelor, determinare (SMIL), 516-517
sfârșitul implicit al elementelor, determinare (SMIL), 514-515
SGML (Standard Generalized Markup Language), 17, 28-29
excluderi, 675
ISO, sit Web, 613-614
privire generală, 17-18
short (întregi), grupul de lucru XML Schema al W3C, 160
Siegel, David, 258
Signal 9 Solutions, ConSeal Private Desktop, 359
Signed Markup Description Language (SMDL), 371-374
SIM SGML Database Technology, sit Web, 619
Simple API for XML. *Vezi și* SAX
Simple Key-management for IP (SKIP), 368
Simple Object Access Protocol. *Vezi și* SOAP
sincronizare, 501-502. Vezi și SMIL
elemente (SMIL)
par, 508-509
seq, 509-510
sincronizarea timpului. *Vezi și* sincronizare
sindromul NIH (not invented here – nu a fost inventat aici), 106
sintaxa de clasă cu punct, 288
sintaxă
CSS, 284-285
DTD, 63
entități, 67
ID-uri HTML/XML, 291
notații, 64
sistem
atribute de test (SMIL), 519-520
conversii ERP, 475
sisteme de întreprindere (ERP), 456-457
Sisteme de tranzacții B2B (business to business), 60
situl Web al Microsoft LinkExchange (profiluri utilizator), 343
situl Web al Microsoft
Distributed Internet Architecture (DNA), 407-408
Internet Explorer 5, 406
Personal Web Server (PWS), 406
Windows NT Server, 406
situri Web
<FONT FACE> considerat dăunător, 258
4DOM tools, 682-683
4XPath tools, 683
4XSLT tools, 683
Academia Sinica Computing Centre, 623
Acord, 615
Adequate.com, 472
Aegis, 259
AIML, 616
Amaya editor, 680
Amaya, 680, 686
AML, 616
analizor Ælfred, 679
analizor fxp, 679
Annotated XML Specification, 620
ANZLIC, 616
Apache XML Project, 624
APPEL, 620
Apple, 504
Associating Style Sheets with XML documents Version 1.0, 620
BAAN, 454
BBB (Better Business Bureau) Online (profiluri utilizator), 342
Bibliomania, 128
Bio.Perl, 616

BIOML, 616
BIPS, 616
BizTalk, 59
Breeze Commerce Studio, 686
BSML, 616
C2net, 367
Café Con Leche: XML News and Resources, 624
Canonical XML, 621
Cayman Systems, 355
CDATA content, 284-285
CERT (Computer Emergency Response Team), 369
CFML, 616
CGI Environment variables, 384
Chami.com, 37, 676
Chinese XML Now! Site, 623
CIDS, 616
Cisco Systems, 355
CiTEC, 279
CML, 576, 616
Cocoon, 681
Coins: Tightly Coupled JavaBeans and XML Elements, 616
Commerce XML, 616
CommerceNet Home Page, 616
Common Markup for micropayment per-fee-links, 621
Corpus Legis Project, 616
CRDP, 616
Cryptography Research, 369
CSS, 283-284
CSS1, 620
CSS2, 620
Customer Support Consortium, 616
Data Documentation Initiative, 616
David Raggett's HTML Tidy, 676, 682
David Siegel, 258
dbXML, 445
DCE/RPC, 432
DDML/Xschema, 616
developerWORKS, IBM, 623
Digital Receipt Infrastructure Initiative, 617
DISA, 617
Distributed Internet Architecture (DNA), 407-408
DMI, 617
DMTF Common Information Model, 617
DMTF Web-based Enterprise Management (WBEM), 617
DocBook, 614-615
DOM Level 1, 621
DOM Level 2, 621
DTD.com, 624
EAD, 617
ECIX, 617
EEMA EDI/EC Work Group: UN-XML, 617
Electronic Frontier Foundation, 369
ERP Hub, 455
EUROMATH Project, 617
European XML/EDI Pilot Project, 617
Exchange Table Model, 617
Extensibility XML Authority, 683
FOP tools, 683
Fortify, 367
Frontier 5 and XML, 617
GCA, 617
GedML: Genealogical Data in XML, 617
Geometry Technologies, 575
Georgia State University Electronic Court Filling Project, 617
GNOME, 624
Health Level Seven, 617-618
Henter-Joyce (cititorul de ecran JAWS), 297-298
HR-XML Consortium, 618
HTML 4.0, 621
HTML 4.01, 621
HTML-kit from Chami.com, 682
HTTP Distribution and Replication Protocol (DRP), 618
IBM alphaWorks Tool Suite, 683-685
IEEESTD V3.0 DTD, 618
IMS, 617
InDelv XML Client, 680
InDelv, 267
Information & Content Exchange (ICE), 618
Interactive Financial Exchange (IFX), 618
interMute, 369
International Layout in CSS, 621
Internet Junkbuster Proxy, 369
iPlanet Web Server, 381-382
ISO, SGML, 613-614
ISO/IEC FCD 13250:1999 - Topic Maps, 621
ISO-12083 standards, 613-614
Situl XMLXperts, 614
James Clark XT, 397
Java (Sun), 436
Java SAX, implementare, 229
Java SDK (Software Development Kit), 229
Java Web Server, 381
JSML, 618
JUMBO, 680
Juniper Networks, 355
JWSDK, 382-383
Lark parser, 682
Macsyma, 575
Manex, 455
Mapics, 454

MARCXML, 615
MathCAD, 575
Mathematica, 685
MathML, 621
MathType, 685
Megginson, David (situl SAX), 230
Meta Content Framework (MCF), 618
Microsoft
  Internet Explorer 5, 406
  Personal Web Server (PWS), 406
  Windows NT Server, 406
Microsoft Internet Explorer, 624
Microsoft LinkExchange (profiluri utilizator), 343
Microstar FastTAG, 682
Mortice Kern Systems, 377
Mozilla, 264, 624
Muffin, 369
Mulberry Technologies, 282
Multidoc Pro Publisher, 681
Navy CALS DTDs, 613
Near & Far Designer, 681
Netopia, 355
Network ICE, 359
Newspaper Association of America (NAA), 618
OASIS, 59, 615
Online Computer Library Center, 465
Ontology Markup Language, 618
Open Content Syndication (OCS), 618
Open Software Description Format (OSD), 618
Open Trading Protocol (OTP), 618
OpenMLS, Real Estate DTD Design, 618
OpenTag, 618
Opera (profiluri utilizator), 335
Opera Software, 262
Oracle, 454-455
P3P standards (profiluri utilizator), 340
P3P, 621
Paged Media Properties for CSS3, 621
Panorama SGML Viewer, 680
Perl XML, 624
Personal Web Server, 380
PICS Signed Labels Personal Web Server (PWS), 621
PivotPoint, 454
Platform for Internet Content Selection (PICS), 618
Plume, 680
Poet Content Management Suite tools, 683
Pretty HTML program, 483
PrivaSeek (profiluri utilizator), 333
Process Interchange Format XML (PIF-XML), 618
programul Tidy, 482
Project Gutenberg, 128
PURL, 128
Python and XML Processing, 681
Python SAX Implementation, 229
Python, 624
RDF Model and Syntax, 621
RDF Schema, 621
RealNetworks, 503
Recomandarea XML 1.0, 646
RFC-2112, 621
Rich Site Summary (RSS), 618
Roman Czyborra, 45
RosettaNet, 619
Ruby, 621
Sable, Speech Synthesis Markup Language, 619
SAE J2008, Automotive Industry Standard, 619
Safe for Kids, 472
Safe Surf, 472
SAX, 681
Signal 9 Solutions, 359
SIM SGML Database Technology, 619
SmartX, 619
SMIL Boston, 622
SMIL, 622
SMPTE (Society of Motion Picture and Television Engineers), 512-513
SOAP, 620
SSML, 619
Steganography Info and Archive Pages, 372
String Identity/Matching, 622
Style One, 682
Sun, 230
SVG, 622
Symantec, 359
TalkML, 619
TclXML, 678, 681
TEI, 128, 614
Telecommunications Interchange Markup (TCIF/IPI), 619
Texcel Information Manager, 445
Text Encoding Initiative Consortium, 185
Theological Markup Language (ThML), 619
Tim Bray's Annotated XML Recommendation, 31-32
Tomcat, 383
Translation Memory eXchange, 619
TRUSTe (profiluri utilizator), 340, 342
TUCOWS, 502
TV Guide (profiluri utilizator), 333
UIML, 619
UML eXchange Format (UXF), 619
Unicode in XML, 622
Unicode-style regular expressions, 147
UPnP Consortium, 439
URIs article, 652
User Agent Accessibility Guidelines 1.0, 622
User interface for CSS3, 622
Vancouver Webpages Rating Service, 471-472
Verso, 281
Virtual Hyperglossary (VHG), 619
Visual SlickEdit, 489
Visual XML, 681
VoiceXML, 619, 622
W3C set de validare online WebLint, 488
W3C XML 1.0 Recommendation, 639
W3C XPath Recommendation, 172
W3C, 145-146, 158, 709-710
WAI, 622
Waterloo Maple, 575
Weather Observation Markup Format (OMF), 619
Web Characterization, 622
Web Content Accessibility Guidelines 1.0, 622
Web Review, 262
Web site privacy statements (profiluri utilizator), 343
Web Standards Project, 406, 619
Web-based Distributed Authoring and Versioning (WEBDAV), 619
WebCGM Profile, 622
WebEQ, 685
Wireless Markup Language (WML), 620
Wolfram Research, 569
Women Writers project, 128
WRQ, 359
Xalan tools, 683
Xerces-J validator, 678
XFDL, 620
XHTML 1.0, 622
XHTML technical reference, 652
XLF Initiative, 620
XLink, 623
XMetaL 1.2, 681
XML <PRO>, 681
XML 1.0, 622
XML Canonicalization, 622
XML Court Interface (XCI), 620
XML Fragment Interchange, 622
XML Hack: Developer news from the XML Community, 624
XML Information Set, 622
XML NITF, 620
XML Schema Part 1: Structures, 622
XML Schema Part 2: Datatypes, 623
XML schema standards updates, 145-146
XML Signature, 623
XML Software Site, 230
XML Spy editor, 679
XML Spy, 155, 310-311, 681
XML Testbed, 681
XML Toolkit, 624
XML, 624
XML.org, 623
XML/EDI Group, 620
XML/EDI Repository, 620
XML4J, analiyor XML Java, 382
XMLephant, 624
XMLNews, 624
XML-RPC Home Page, 620
XMLSoftware, 624
XMLTP interprocess communications, 435
XMLwriter editor, 679
XMLwriter, 681
XP parser, 681
XPath Version 1.0, 622
XPointer, 622
XSL Specification, 621
XSLT Version 1.0, 623
XT tools, 682
zone de nume
  ISBN-uri, 139-140
  ISSN-uri, 139
  limbi non-europene, 137-138
  PURL-uri, 141-142
  XML, 621
SKIP (Simple Key-management for IP), 368
SmartX, sit Web, 619
SMDL (Signed Markup Description Language), 371-374
SMIL (Synchronized Multimedia Integration Language), 59, 500-501, 622
  atribute de test, 519-520
  capacitățile dispozitivului de ieșire, 536
  depanarea, 537
  documente
    antetul documentului, 504-506
    corpul documentului, 505-507
    creare, 502-503
  elemente
    a, 507, 520-521
    anchor, 521
    animation, 507
    audio, 507
    body, 507-508
    elemente obiecte multimedia, 510-511
    head, 504
    hiperlegare, 520-521
    img, 507
    layout, 504-505
    meta, 506

par, 507-509
ref, 508
region, 505-506, 505-506
root-layout, 506
seq, 508, 509-510
smil, 504
switch, 508, 518-519
text, 508
textstream, 508
video, 508
implementări în lumea reală, 503
Apple Quicktime, 504
RealSlideshow, 503
limbajul de machetare de bază, 505
modificările utilizatorilor, 536-537
prezentări fotografice, 521-522
prezentări multimedia, 522
problema calculatoarelor portabile, 534-535
sincronizarea, 501-502
sit Web, 622
tehnologiile viitorului, 536-537
tipuri multimedia, 524-525
audio, 525-528
coordonarea cu instrumentele multimedia existente, 528
legende, 528-530
narațiunea, 528-530
secvențe audio descriptive, 531-534
text, 528
video, 528
valorile modelului temporal, 512
determinare, 513-518
NPT (Normal Playing Time), 512
SMPTE (Society of Motion Picture and Television Engineers), 512-513
SMIL Boston, sit Web, 622
SMPTE (Society of Motion Picture and Television Engineers), 512-513
SOAP (Simple Object Access Protocol), 438-439, 620
firewall-uri, 438-439
protocoale de date bazate pe XML, 440-441
sit Web, 620
tranzacții, 440
UpnP, 439
Society of Motion Picture and Television Engineers (SMPTE), 512-513
socluri
SSL (Secure Sockets Layer), 374-375
XML, 419
software
Academia Sinica Computing Centre, 623
Apache XML Project, 624
Café Con Leche: XML News and Resources, 624
Chinese XML Now! Web Site, 623
cititorul de ecran JAWS, 298
developerWORKS, IBM, 623
DTD.com, 624
GNOME, 624
Java, 379-380
Microsoft Internet Explorer, 624
Mozilla, 624
Perl XML, 624
platforma Psion, 301
Python, 624
XML Hack: Developer news from the XML Community, 624
XML Toolkit, 624
XML, 624
XMLephant, 624
XMLSoftware, 624
sortare (XSL), 321
SOX (Schema for Object-Oriented XML), 167
scheme XML, 167
spații albe, 51-52
CSS, 730
mecanismul de recuperare HTML, 486
specificare
date prestabilite, 76
versiuni, declarații, 646
Specificația Extensible Stylesheet Language. *Vezi și* XSL
Specificația XML 1.0 adnotată, sit Web, 620
Speech Synthesis Markup Language (SSML), 619
spoofing, 353-354
SSL (Secure Sockets Layer), 374-375
SSML (Speech Synthesis Markup Language), 619
sit Web, 619
Standard Generalized Markup Language. *Vezi și* SGML
standarde deschise (Microsoft), 406-407
standarde, standarde deschise (Microsoft), 406-407
steganografie, 372
Steganography Info and Archive Pages, 372
stiluri, reguli, 287
stive, 350
stocarea datelor, 442
RTF, 442
strategii (ERP), 453-456
String Identity/Matching, sit Web, 622
Stronghold, 367
structura unui atribut XML Schema, 158
structura unui element XML Schema, 157-158
structură (XML), 15-16, 30-31
excludere, 15-16
structură fizică, 49-50
structură logică, 48-49
structuri, servlet-uri, 392-393
Style One, instrumente, 682
sit Web, 682
submulțimea internă a DTD-ului, 67
Sun Java, situri Web, 436
Sun, sit Web, 230
sunet (CSS), 739-743
sunet surround, 300
suport pentru tipul de mediu al agenților utilizator, 291
suportul pentru seturi de caractere (HTML), 485
suprascrierea zonelor de nume, 137
SVG (Scalable Vector Graphics), 622
Requirements, situl Web, 622
Synchronized Multimedia Integration Language. *Vezi și* SMIL

Ș

șabloane (XSL), 305-311
generarea de conținut, 320
șir comentariu SGML (<!— —>), 145
șir comentariu SGML/XML
(<!—), 144
(—>), 144
șiruri comentariu
JavaScript, //, 145
SGML/XML (<!—), 144
șiruri
grupul de lucru XML Schema al W3C, 159
selectare, 204

T-Ț

tabele
CSS, 734-737
entități W3C, 88
MathML, 552-554
tehnici de afișare pentru vedere slabă, 593-595
TalkML, sit Web, 619
TclXML
analizor, 681
sit Web, 678-681
validator, 678-679
TCP (Transmission Control Protocol), 352
stiva protocolului, 350-351
criptare, 368
IP (Internet Protocol), 351
pachete, 351
tehnici de afișare pentru vedere slabă, 588-589
cadre, 591-592
conținut plug-in multimedia, 593
fișiere PDF (Portable Document Format), 592
fundaluri, 589
legături, 590-591
medii de redare JavaScript, 592
notații pentru imagini, 589-590
tabele, 593-595
text, 589
tehnologie pointer, 58
TEI (Text Encoding Initiative), 614
DTD-uri, 614
pointeri, 198
sit Web, 128, 185, 614
Telecommunications Interchange Markup (TCIF/IPI), sit Web, 619
telefoane mobile, afișaje, 301
telefonul mobil NeoPoint 16000, mod dual, 301
testare
Active Server Pages (ASP-uri), 411-412, 420-421
DTD-uri, 112-113, 118-120
foi de stil, 274
validitatea codului sursă, 101-102
Texcel Systems din Norvegia, 445-447
Information Manager, situri Web, 445
text
#CDATA, 144
#PCDATA, 144
arbitrar, evitarea entităților parametru, 125-126
casete cu citat evidențiat, 293
CSS, 721-723
tehnici de afișare pentru vedere slabă, 588-589
tipuri de mediu SMIL, 528
Text Encoding Initiative. *Vezi și* TEI
Theological Markup Language (ThML), sit Web, 619
Tim Bray's Annotated XML 1.0 Recommendation, 31-32
time, grupul de lucru XML Schema al W3C, 160
timeDuration, grupul de lucru XML Schema al W3C, 160
timeInstant, grupul de lucru XML Schema al W3C, 160

timePeriod, grupul de lucru XML Schema al W3C, 160
timp de încărcare (DTD), 30-31
tip de conținut text/html (ASP), 411-412
tip de conținut text/plain (ASP), 411-412
tip de conținut
  ASP, 411-412
  cod CGI, scriere, 388
  servlet-uri, 395
tipărire sugestivă, 521. *Vezi și* înfrumusețare
tipul de mediu aural (auditiv), 292
tipul de mediu Braille, 292
tipul de mediu embossed, 292
tipul de mediu handheld, 292
tipul de mediu print, 292
tipul de mediu projection, 292
tipul de mediu screen, 292
tipul de mediu tty, 292
tipul de mediu tv, 292
tipul media all, 292
tipuri de medii, 292
  (CSS2), 291
  CSS2, 297
  extindere, 302
  proprietăți auditive, 298-299
  sensibilitatea la tipul literelor, 292
  SMIL, 524-525
    aspecte legate de calculatoarele mobile, 534-535
    audio, 525-528
    coordonarea cu instrumentele multimedia existente, 528
    dispozitive de ieșire, 536
    legende, 528-530
    narațiuni, 528-530
    secvențe audio descriptive, 531-534
    text, 528
    video, 528
  sunet surround, 300
Tomcat, sit Web, 383
Topic Maps - ISO/IEC FCD 13250:1999, sit Web, 621
TopStyle
  instrument, 273-274
  validator de foi de stil, 297, 303
transcludere, 185
  XLink, 186
    în fișiere text multimedia, 185
    legături externe, 197
    legături extinse, 186-189
    legături in-line, 196-197
    legături simple, 186-187, 194-195
transferul de informații (HTTP), 402
transformarea XML (HTML), 397-399
transformarea XML în HTML pe server, 412-415
Translation Memory eXchange, sit Web, 619
Transmission Control Protocol (TCP), 352
tranzacții (SOAP), 440
tratarea excepțiilor (DOM), 224
tratarea excepțiilor (DOM), 224
trimitere de formulare, metoda doGet(), 393
TRUSTe (profiluri utilizator), sit Web, 340, 342
TUCOWS, sit Web, 502
tunneling, 368
Tunnicliffe, William, 28
Turing, Alan, 364
TV Guide (profiluri utilizator), sit Web, 333
ținte, 291

U

UIML (User Interface Markup Language), 619
  eXchange Format (UXF), sit Web, 619
  sit Web, 619
unghiuri, 300
Unicode in XML, sit Web, 622
Unicode-style regular expressions, sit Web, 147
Uniform Resource Name (URN), 475
unități de măsură, 300
unități de măsură (CSS), 743-744
unități de măsură pentru frecvența auditivă, 300
Universal Plug and Play. *Vezi și* UPnP
unsignedByte, grupul de lucru XML Schema al W3C, 160
unsignedInt, grupul de lucru XML Schema al W3C, 160
unsignedLong, grupul de lucru XML Schema al W3C, 160
unsignedShort, grupul de lucru XML Schema al W3C, 160
UPnP (Universal Plug and Play), 439
  situri Web ale Consorțiului, 439
uri-reference, grupul de lucru XML Schema al W3C, 161
URIs article, sit Web, 652
URL-uri (CSS), 743-744
urmărirea poziției în documente (XPath), 172-174

URN-uri, 475
  CSS, 743-744
  declararea zonelor de nume, 139-140
User Agent Accessibility Guidelines 1.0, sit Web, 622
User Interface Markup Language (UIML), 619
utilizatori
  coregrafia XML, 627-628
    multiple, 634-635
  profiluri, 333
    asistenți de confidențialitate, 343-345
    baze de date server-side, 335-336
    BBBOnline, sit Web, 342
    brokeri de confidențialitate, 336
    cookie-uri, 334-335
    declarații de confidențialitate, sit Web, 343
    ID-uri/parole utilizator, 336-337
    Microsoft LinkExchange, sit Web, 343
    negociere, 333-334
    Opera, sit Web, 335
    OPS/P3P, 341
    P3P Standards, sit Web, 340
    PrivaSeek, sit Web, 333
    TRUSTe, sit Web, 340
    TV Guide, sit Web, 333
utilizatori cu handicap. *Vezi și* accesibilitate
utilizatori cu handicap auditiv, legende, 528-530
utilizatori cu handicap fizic. *Vezi și* accesibilitate
utilizatori cu handicap vizual
  audio descriptiv, 531-534
  narațiune, 528-530
  redare auditivă
    editare vocală, 599
    foi de stil auditive, 596-599
    medii țintă multiple, 599
  tehnici de afișare pentru vedere slabă, 588-589
    cadre, 591-592
    conținut multimedia plug-in, 593
    dispozitive de ieșire Braille, 595-596
    fișiere PDF(Portable Document Format), 592
    fundale, 589
    imagini, 589-590
    legături, 590-591
    medii de redare JavaScript, 592
    notații image, 589-590
    tabele, 593-595
    text, 589
UUID-uri Registry, modelul de obiect COM, 432-433

V

validare
  conținutul elementului (scheme XML), 163
  scheme (XML Spy), 155-156
  structura documentului, 14
  XHTML din HTML, 484-490
validatoare
  Ælfred, 679
  fxp, 679
  TclXML, 678
  Xerces-J, 678
validatorul Ælfred, 679
  sit Web, 679
validatorul fxp, 679
  sit Web, 679
validatorul Xerces-J, 678
  sit Web, 678
validitate, 640
  cod sursă, testare, 101-102
  completitudinea înregistrărilor, limitări, 164-165
  constrângeri (atributul standalone), 646
valori
  apostrofuri și ghilimele, 37
  atribute (HTML), 484
  locale, furnizarea lor la DTD-uri externe, 108-109
valori atribut
  HTML, 483-484
  apostrof și ghilimele, 37
valori locale, DTD-uri externe, 108-109
valori Normal Playing Time (NPT), 512
valori pentru măsurarea timpului, 300
valori prestabilite, setare scheme XML, 167
valorile modelului temporal (SMIL), 512
  determinarea, 513
    începutul dorit al elementelor, 516
    începutul efectiv al elementelor, 516
    începutul implicit al elementelor, 513-514
    sfârșitul dorit al elementelor, 515
    sfârșitul efectiv al elementelor, 516-517
    sfârșitul implicit al elementelor, 514-515
  NPT (Normal Playing Time), 512
  SMPTE (Society of Motion Picture and Television Engineers), 512-513
Vancouver Webpages Rating Service, 471-472
variabila XMLNS, 252
variabile de mediu CGI, sit Web, 384
variabile de mediu, calea de includere a claselor, 404

verificarea completitudinii înregistrărilor (scheme XML), 163
verificarea legăturilor, 652
verificatoare ortografice, 423-424
versiuni, declarații de specificare, 646
Verso, sit Web, 281
vid
conținut, elemente, 73-74
elemente, 650
etichete, 48-51
video, tip de mediu SMIL, 528
viermi (Internet Worm), 352-353
viitorul
SMIL (Synchronized Multimedia Integration Language), 536-537
XML, 24
XLink, 25
XPointer, 25
XSL (Extensible Stylesheet Language), 25-26
Virtual Hyperglossary (VHG), sit Web, 619
Virtual Private Networks (VPN-uri), 360
Virtual Reality Modeling Language. ***Vezi și*** **VRML**
Visual Basic
cod, depanare, 421-422
obiecte COM
arbori de ieșire XML, construire, 418-419
condiții de eroare, verificare, 418
creare, 416-420
DLL-uri, creare, 416-420
prelucrarea XML, 418
Visual SlickEdit, 135-136
sit Web, 489
vizualizarea logică
arborele documentului DOM Level 2, 213-215
filtre, 214-215
indicatoare whatToShow, 213-214
vizualizări, vizualizarea documentului, 210
vocabulare, 32
AIML, 580
AML, 580
CML, 540, 576-580
Dublin Core, 460-461, 464-470, 473
IML, 580
MathML, 539-540
algebră liniară, 564
algebră, 560-561
aritmetică, 560-561
browserul Amaya, 569
calcule, 561-562
combinarea de matematică și text, 568-569
combinarea de prezentări și elemente de sintaxă, 566-567
conversii, 542
editorul de ecuații WebEQ, 575
elemente de conținut. *Vezi și* elemente de sintaxă de mai jos
elemente de prezentare versus elemente de sintaxă, 566-567
elemente de prezentare, 542-545
elemente de sintaxă, 556-565
elementul de interfață, 565-566
importanța, 540-542
limitări, 539-540
listinguri, 569-575
logică, 560-561
Macsyma, 575
Maple, 575
MathCAD, 575
Mathematica, 569-575
mulțimi, 562
recomandări, 539-540, 544
referințe, 539-540
relații, 561
reprezentare a semnificațiilor, 564
scheme de machetare, 547-554
specificația MathML 2.0, 555-556, 565
statistică, 563-564
șiruri și serii, 562-563
transferabilitate între medii, 539-540
trigonometrie, 563
zonă de nume, 542
PICS, 460-461, 470-471
Adequate.com, 472
RSACi, 471
Safe for Kids, 472
Safe Surf, 472
Vancouver Webpages Rating Service, 471-472
RDF, 460-463
colecții, 477-478
definirea vocabularelor RDF universale, 464-465
depanare, 479-480
funcționare, 462-463
proprietăți, 475-476
resurse, 475
sintaxă, 474
utilizare, 478-479
valori Dublin Core, 473
valori, 477
VoiceXML, 526-528
sit Web, 619-658
VoxML, 526
VPN-uri (rețele locale private), 360
VRML (Virtual Reality Modeling Language), coregrafia XML, 633-634
vulnerabilitatea datelor, 347-349. ***Vezi și*** **securitate**

W

W3C (Consorțiul World Wide Web)
CSS, 283-284, 710-711, 748
DOM (Document Object Model), 208
grupul de lucru XML Schema
binary, 161
boolean, 159
byte, 160
century, 161
date, 160
decimal, 159
double, 159
entities, 161
entity, 161
float, 159
id, 161
idrefs, 161
int (întregi), 159
integer, 159
language, 161
long (întregi), 159
month, 160
name, 161
ncname, 162
negativeInteger, 160
nmtoken, 161
nmtokens, 161
nonNegativeInteger, 160
nonPositiveInteger, 160
notation, 161
notations, 161
positiveInteger, 160
qname, 161
recurringDate, 160
recurringDay, 160
recurringDuration, 160
short (întregi), 160
string, 159
time, 160
timeDuration, 160
timeInstant, 160
timePeriod, 160
unsignedByte, 160
unsignedInt, 160
unsignedLong, 160
unsignedShort, 160
uri-reference, 161
year, 160
organisme standard
Annotated XML Specification, 620
APPEL, 620
Associating Style Sheets with XML documents Version 1.0, 620
Canonical XML, 621
Common Markup for micropayment per-fee-links, 621
CSS1, 620
CSS2, 620
DOM Level 1, 621
DOM Level 2, 621
HTML 4.0, 621
HTML 4.01, 621
International Layout in CSS, 621
ISO/IEC FCD 13250:1999 - Topic Maps, 621
MathML, 621
Namespaces in XML, 621
P3P, 621
Paged Media Properties for CSS3, 621
PICS Signed Labels, 621
RDF Model and Syntax, 621
RDF Schema, 621
RFC-2112, 621
Ruby, 621
SMIL Boston, 622
SMIL, 622
String Identity/Matching, 622
SVG Requirements, 622
SVG, 622
Unicode in XML, 622
User Agent Accessibility Guidelines 1.0, 622
User Interface for CSS3, 622
VoiceXML, 622
WAI, 622
Web Characterization, 622
Web Content Accessibility Guidelines 1.0, 622
WebCGM Profile, 622
XHTML 1.0, 622
XLink, 623
XML 1.0, 621
XML Canonicalization Requirements, 621
XML Fragment Interchange, 621
XML Information Set, 621
XML Schema Part 1: Structures, 621
XML Schema Part 2: Datatypes, 622
XML Signature, 623
XML.org, 623
XPath Version 1.0, 622
XPointer, 622
XSL Specification, 621
XSL, 623
XSLT Version 1.0, 623
Recomandarea XML 1.0
entități, 647-648
sit Web, 639
Recomandarea XPath, 172
servicii de validare, 710
sit Web, 213, 488
tabelul de entități, 88
entități parametru, 89
WAI (Web Accesibility Initiative), 622
sit Web, 622
Waterloo Maple, 575

Weather Observation Markup Format (OMF), sit Web, 619
Web
căutare
căutări independente de context, 463-464
PICS, 470-472
RDF, 462-463
evaluarea siturilor Web, 470-472
Web Accesibility Initiative. ***Vezi și*** **WAI**
Web Characterization, sit Web, 622
Web Content Accessibility Guidelines 1.0, sit Web, 622
Web Review, sit Web, 262
Web Standards Project (WSP), 619
Web Standards Project, sit Web, 406
Web-based Distributed Authoring and Versioning (WEBDAV), sit Web, 619
WebCGM Profile, sit Web, 622
WebEQ, aplicație cu suport XML, 685
sit Web, 685
Windows DNA, 407-408
Wireless Markup Language (WML), sit Web, 620
Wolfram Research, 569
Women Writers project, sit Web, 128
World Wide Web
căutare
căutări independente de context, 463-464
PICS, 470-472
RDF, 462-463
evaluarea siturilor Web, 470-472
WRQ AtGuard, 359

X

XFDL (Extensible Forms Description Language), sit Web, 620
XHTML (Extensible HyperText Markup Language, 19, 34-35, 37, 481-482. ***Vezi și*** **HTML (HyperText Markup Language), 19**
casete de text mobile simulate, 295
conținut de stil, 284-285
depanarea incompatibilității browserului, 496, 498
DTD-uri
documente XML, redare, 109
etichete ancoră, 676
extensibilitate, 677
încorporarea de elemente, 99-101
sensibilitate la tipul literelor, 675
zone de nume, 677
etichete, 109, 134-136
extensibilitate, 486
preprocesare particularizată, 487-488
probleme de imbricare, 487
foi de stil, 284-285
HTML-Kit / Tidy, 676
legături externe, 284-285
personalizare, 490
prelucrarea de stil, 277-279
privire generală, 19
seturi de caractere
entități caracter (ordonate alfabetic), 698-703
entități caracter (ordonate numeric), 703-708
entități caracter matematice, grecești și simbolice ISO, 693-698
entități caracter speciale ISO, 692-693
entități ISO Latin-1, 688-689, 691
sit Web, 652
transformarea din HTML, 489-494, 496
translatarea din minimizarea HTML, 482-483
validarea din HTML, 484-488, 490
zone de nume, 130-133
XHTML 1.0, sit Web, 622
XLF Initiative, sit Web, 620
XLink, 25, 57, 184-185, 304
atribute, 189, 191
actuate, 190
arc, 190
auto, 190
extended, 189
from, 190
href, 189
locator, 189
new, 190
none, 190
parse, 190
replace, 190
resource, 190
role, 191
show, 190
simple, 189
title, 190, 191
to, 190
type, 189
user, 190
comportamentul legăturii, 198
depanare, 211
legături întrerupte, 204
sintaxa de legatre, 204
etichete ancoră, legare, 183
legare, 185
sit Web, 622
structuri de elemente, 191
elemente arc, 193
elemente extinse, 192-193

elemente locator, 193
elemente resource, 193-194
elemente simple. 198-199
elemente title, 194
transcludere, 185
în fișiere text multimedia, 185
legături externe, 197
legături extinse, 186-189, 196
legături in-line, 196-197
legături simple, 186-187, 194-195
XLL (XML Linking Language)
XLink, 25
XPointer, 25
XMetaL 1.2, instrumente de dezvoltare XML, 681
sit Web, 681
XML (Extensible Markup Language)
<PRO> XML, instrumente de dezvoltare, 681
sit Web, 681
analizoare
MSXML, 406
tipuri de elemente, 52-53
avantaje
dezvoltarea Web, 15
flexibilitatea machetelor, 14
independența de platformă, 14
precizie, 14
proiectare orientată spre obiecte, 15
validarea structurii documentului, 14
browsere cu suport XML, 284-285
caractere de marcare, EBNF, 645-646
clase de atribute, 290-291
comentarii, 284-285
construit corect, 83-84
conținut, înglobarea etichetelor XHTML, 109
convertirea din HTML, 290-291
CORBA, 404
coregrafia, 626-627
alfabete multiple, 632
Comportamente Microsoft DHTML, 633
depanare, 636-637
fluxuri de date multiple, 634-635
limbi naturale, 627
limbile Microsoft Internet Explorer, 631
limbile Mozilla, 629-631
limbile Opera, 632
localizare, 628-629
mai multe limbi, 628
mai multe medii, 632
mai mulți utilizatori, 635
medii interactive, 632-633
Microsoft Chrome, 634
Sistemul de acțiune în cascadă Netscape, 633
suport pentru limbi, 629-631
utilizatori, 627-628
viitorul HTML, 636
VRML, 633-634
depozite de date, integrare, 441
dezvoltări viitoare, 24
XLink, 25
XPointer, 25
XSL (Extensible Stylesheet Language), 25-26
documente, 47
constrângeri, 38-42, 52-53
construire corectă, 38-42, 52-53
corp, 505-507
creare, 39
date analizate, 38
date caracter, 47-48
date neanalizate, 38
declarații, 43
Document Type Declaration, 46-47
DTD-uri XHTML, redare, 109
marcaje, 48-56
ținte, 291
DTD SGML, convertire, 97
DTD-uri (Document Type Definitions), 24
EBNF (Extended Backus-Naur Form), 31-32
elemente
ca selectori, 286
redarea proprietăților, 290
valori pentru proprietăți, 288
etichete, ordinea lor, 21-23
exemplu practic, 20-21
fișiere, editare, 135-136
foi de stil, 284-285
gramatica EBNF, listing, 640-645
granulozitatea și structura datelor, 15-16
descendenți, 16-17
excludere, 15-16
HTML, 34-35, 37
transformare, 397-399
ID-uri, 288, 291
instrucțiuni de prelucrare, 284-285
integrarea în ERP, 449-450
istoria, 26
conceptul Generic Coding, 28
convenții de marcare în ebraica veche, 27
convenții de marcare în latina veche, 27-28
GML (Generalized Markup Language), 28
inițiative recente, 28-29
Runoff, 28
SGML (Standard Generalized Markup Language), 28-29
înregistrarea clienților, listing, 153-156
prelucrarea de stil, 277-279
privire generală, 12, 17, 19-20

procesoare
CDATA, 52
entități neanalizate, 56-58
linii noi, 51
spații albe, 51-52
profiluri de document, 490
avantaje, 490
Referință Tehnică, 639
referințe pentru foi de stil, 396
schema standards updates, sit Web, 145-146
scheme
analiză, 146
atribute locale, 152-153, 155-156
câmpuri de intrare cu format liber, 165
conlucrarea între aptitudini, 148
constrângerea conținutului elementului și atributului, 162-163
definire, 152
definirea atributelor, 158
definirea elementelor, 157
definirea modelelor de conținut, 162
definirea tipurilor de date, 158-161
extinderea tipurilor de date, 162
identificarea compromisurilor, 147-148
limitările verificării erorilor și a validității, 164-165
listinguri, 307-308
modificări în, 145-146
privire generală, 143
propuneri anterioare, 166
punctele forte, 146-147
RDF, 166
setarea valorilor prestabilite, 167
SOX, 167
structura elementelor, listing, 151-152
structura unui atribut, listing, 158
utilizarea zonelor de nume, 148-151
validarea câmpurilor definite prin scheme, 150
validarea conținutului elementului, 163
verificarea caracterului complet al înregistrărilor, 163
XML Schema Part 0: Primer (abecedar), 148
XML Schema Part 1: Structures (structuri), 148
XML Schema Part 2: Datatypes (tipuri de date), 148
sensibilitate la tipul literelor
servlet-uri, cerere, 399-402
sintaxa de clasă cu punct, 288
sit Web, 681
situri Web, 620, 623-624
socluri, 402
Spy, 310-311
instrumente de dezvoltare, 680-681
înregistrare clienți validată de scheme, 155-156
sit Web, 155, 680-681
structură, 30-31
structura fizică, 49
structura logică, 48-49
suport pentru limbi, 37
Testbed XML, instrumente de dezvoltare, 681
Tim Bray's Annotated XML 1.0 Recommendation, 31-32
Toolkit, sit Web, 624
valid, 83-84
vocabulare, 32
XPath (Path Language), 622
XPointer (Pointer Language), 622
zone de nume, 106, 170
XML construit corect, 83-84
caractere, 83
etichete, 83
XML Hack: Developer news from the XML Community, sit Web, 624
XML Linking Language. *Vezi și* XLL
XML Schema Part 0: Primer (abecedar), 148
XML Schema Part 1: Structures (structuri), 148
sit Web, 622
XML Schema Part 2: Datatypes (tipuri de date), 148
sit Web, 623
XML semnat, 370-371
XML Software Site, 230
XML Transfer Protocol. *Vezi și* XMLTP, 434
XML valid, 83-84
XML.org, sit Web, 623
XML/EDI Group, sit Web, 620
XML/EDI Repository, sit Web, 620
XML-Data (scheme XML), propuneri anterioare, 166
XMLephant, sit Web, 624
XMLNews, sit Web, 620
XML-RPC Home Page, sit Web, 620
XMLSoftware, sit Web, 624
XMLTP (XML Transfer Protocol)
brokeri de obiecte, cerere, 426-427
comunicații între procese, 434
depanare, 435
situri Web, 435
XMLwriter
instrumente de dezvoltare, 680-681
situl Web al editorului, 678
XMLXperts (ISO-12083), sit Web, 614
XP
situl Web al analizorului, 681-682
XPath (XML Path Language), 169, 304
abrevieri, 171, 174
exemple, 174-175
lista, 175-177
axe, 169-171
contexte, 170
funcții, 177
boolean(), 179
ceiling(), 179
concat(), 178
contains(), 178
count(), 177
false(), 179
floor(), 179
id(), 177
lang(), 179
last(), 177
local-name(), 177
name(), 177
namespace-uri(), 177
normalize-space(), 178
not(), 179
number(), 179
position(), 177
round(), 179
starts-with(), 178
string(), 178
string-length(), 178
substring(), 178
substring-after(), 178
substring-before(), 178
sum(), 179
translate(), 178
true(), 179
noduri, 171-172
noduri atribut, 171
noduri context, 172
noduri text, 171-172
relații între ele, 172-174
poziția în document, urmărire, 172-174
predicate, 179
Recomandarea XPath a W3C, 172
suport pentru căi de locație, 170-171
suportul XML Namespace, 170
XPath Version 1.0, sit Web, 622
XPointer (XML Pointer Language), 25, 57, 184, 198, 304, 622
adresare absolută, 201
adresare numai prin nume, 184
adresare relativă, 200-201
depanare, 204
legături întrerupte, 204
sintaxa de legare, 204
extensii, 202-203
funcții booleene, 203
funcții pentru locații absolute, 203
end-point(), 203
here(), 203
id(), 203
origin(), 203
start-point(), 203
legături la documente numai pentru citire, 200
selectare după nume, 201-202
sintaxă, 198-199
sit Web, 623
XSchema (scheme XML), 167
XSL (Extensible Stylesheet Language Specification), 25-26, 259, 264-268, 304, 621
arbori de transformare, 326-328
depanare, 282, 329
listinguri, 266-267, 274-277, 323-328
Microsoft, 322
numerotare, 322
obiecte de formatare (fo), 311-312
proprietăți de formatare, 313-320
sit Web, 621
sortare, 321
suport browser, 274-277
șabloane, 305-311
generarea conținutului, 320
transformarea XML în HTML, 412-414
utilizarea cu CSS, 279-280
XSLT (XSL Transformations), 259, 304, 323-328, 623
XSLT Version 1.0, sit Web, 623
xt.jar, 397

Y-Z

year, grupul de lucru XML Schema al W3C, 160
zarf, definiție, 122
zone de nume prefixate, declarare, 131-133
zone de nume prestabilite, 132
zone de nume, 129-130
cu DTD-uri, 129-130
declarare, 130-131
identificate prin contexte non-standard, 140-141
identificate prin ISBN-uri, 139
identificate prin ISSN-uri, 139
identificate prin URN-uri, 139
în limbi non-europene, 137-138
definire, 135-136
DTD-uri XHTML, 676
DTD-uri, 104
compatibilitate, 126-127

identificate prin ISBN-uri, situri Web, 139-140
identificate prin ISSN-uri, situri Web, 139
identificate prin PURL-uri, situri Web, 142
în atribute, identificare, 134-135
în elemente, identificare, 134-135
în limbi non-europene, situri Web, 137-138
MathML, 542
prefixate, declarare, 131-133
prestabilite, 132
scheme XML
utilizare, 148-151
validarea câmpurilor definite prin scheme, 150
suprascriere, 137
XHTML, 130-133
XML (DTD-uri), 106-107
zone de nume, suport pentru XPath, 170

BIBL. CENTR. UNIV. „M. EMINESCU" IAȘI
INFORMATICĂ